Universidad Carlos III de Madrid

Departamento de Ingeniería de Sistemas y Automática

TRABAJO DE FIN DE MASTER

MODOS ALTERNATIVOS DE TRANSPORTE: DYNAMIC RIDESHARING

Máster en Ingeniería Industrial

Autor: Pablo Grandas Aguado
Director: Jorge Villagra Serrano
Tutor: Ramon Barber Castaño

Leganés, septiembre de 2014

Título: MODOS ALTERNATIVOS DE TRANSPORTE: DYNAMIC RIDESHARING
Autor: Pablo Grandas Aguado
Directores: Jorge Villagra Serrano y Ramón Barber Castaño.

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

A todos los que me han ayudado de alguna manera, gracias.

Especialmente a Jorge Villagra por la dedicación en las circunstancias en las que se ha desarrollado el proyecto, al Centro de Automática y Robótica del CSIC por las facilidades, a mi familia y a Elena.

Resumen

El consumo energético, las congestiones de tráfico, las emisiones de contaminantes y los costes económicos son algunos de los factores negativos asociados al transporte privado. Para intentar reducir los efectos de estos se han planteado una serie de modos de transporte alternativos basados en el consumo eficiente de los recursos disponibles. Uno de estos modos o estrategias de transporte es el *dynamic ridesharing.* Este modelo trata de incrementar la tasa de ocupación media de los vehículos privados para, de forma colaborativa, disminuir el número de coches en circulación y por lo tanto los factores citados anteriormente. El *ridesharing* se basa en emparejar a una serie de usuarios con vehículo propio, dispuestos a desviarse de su ruta habitual a cambio de compartir los gastos del viaje, con otros usuarios con una necesidad de transporte. El *dynamic ridesharing* realiza estos emparejamientos en un periodo corto de tiempo, permitiendo a priori así su adopción masiva a través de dispositivos nómadas, como los *smartphones*.

El presente trabajo estudia este modelo de transporte, tanto desde el punto de vista cualitativo, realizando un estudio de la situación actual de este problema, como cuantitativo, desarrollando un modelo matemático del sistema y estudiando una serie de hipotéticos escenarios para comprobar la viabilidad de *ridesharing*. Para esto último es necesario emplear técnicas de programación lineal y optimización no lineal, cuya implementación y evaluación para este tipo de problemas será el objeto de la parte final de este trabajo.

Palabras clave: Modelos de transporte, optimización, programación lineal, algoritmos genéticos, *dynamic ridesharing*.

Abstract

Power consumption, traffic congestion, air pollution and economic costs are some of the negative factors associated with private transport. In order to reduce these effects, different transport modes have raised based on the efficient use of the available resources. One of these modes is the dynamic ridesharing. Dynamic ridesharing tries to increase the average occupation of private vehicles, in a collaborative way, in order to decrease the number of vehicles on road and to mitigate the previously described effects. The ridesharing problem is based on matching drivers, with their own vehicle, with riders and drivers requiring the minimal effort for both, and sharing the costs associated to the ride. Dynamic ridesharing arranges these matches on a very short notice.

In this project we study the proposed transportation mode by analyzing the current state of the art of the problem and developing a mathematical model of the system. This model will be tested on artificial scenarios to prove its feasibility. To achieve such a task, the use of lineal programming and non-lineal optimization techniques will be necessary, being the implementation and evaluation of these algorithms to core work in this project final part.

Keywords: Transport network, optimization, linear programming, genetic algorithms, dynamic ridesharing.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

1. Introducción

El sector energético es el mayor responsable del conjunto de las emisiones, que en 2012 representó el 78% del total. Las emisiones más importantes se deben a la generación de electricidad y al transporte por carretera. Las emisiones en el transporte crecieron un 43,7% entre 1990 y 2012 y suponen el 21,7% del total. En el transporte por carretera las emisiones estuvieron desbocadas hasta 2007, y aunque la crisis económica ralentizó su crecimiento desde 2008, siguen teniendo un gran peso específico.

El sector del transporte es el mayor consumidor de energía y mayor productor de contaminantes en la actualidad. En concreto, tal y como se puede apreciar en la Figura 1, el transporte por carretera consume aproximadamente el 50 por ciento de los derivados líquidos del petróleo [1].

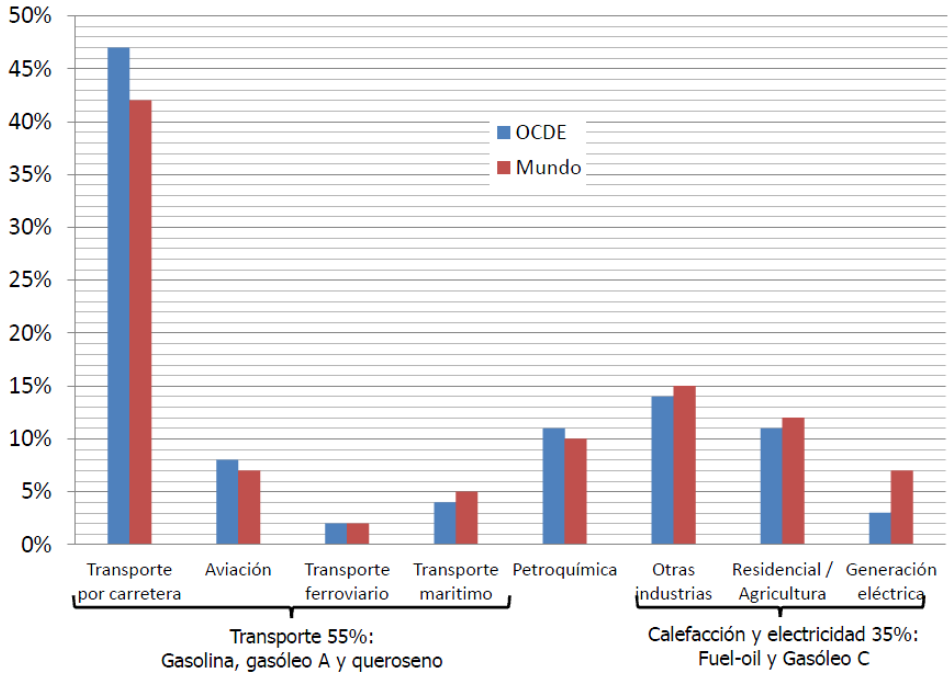


Figura 1.- Reparto del consumo de combustibles derivados del petróleo

Existen estudios recientes [2] que concluyen que en el sector del transporte la reducción de emisiones se debe, sobre todo, al aumento de los precios de los combustibles y a la disminución de desplazamientos laborales y de mercancías los últimos años por efecto de la crisis. En cambio, se siguen incentivando infraestructuras de carreteras de alta capacidad o los aparcamientos para automóviles, al tiempo que se cierran vías de ferrocarril. No se fomenta un cambio modal en el transporte, ni la reducción de los desplazamientos. A todo esto hay que sumarle el gran parque automovilístico (Figura 2) de España, que debido a la crisis se está viendo envejecido y, por lo tanto, volviendo menos eficiente. El ratio de vehículos por persona en España a finales de 2012 era de 2,08 habitantes por vehículo (esta cifra se calcula sobre el total de la población, no únicamente por mayores de edad con facultades para conducir).

TIPOS DE VEHÍCULOS	PARQUE AL 31-XII-2012	DISTRIBUCIÓN PORCENTUAL
Camiones y furgonetas	4.984.722	16
Autobuses	61.127	0
Turismos	22.247.528	71
Motocicletas	2.852.297	9
Tractores Industriales	186.964	1
Remolques y semirremolques	410.369	1
Otros Vehículos (1)	460.196	1
TOTAL (2)	31.203.203	100

Figura 2.- Parque automovilístico de España en 2012. Fuente DGT

Por ello, asociaciones medioambientales [3] apuestan por fomentar una mayor ocupación de los vehículos, tanto en el transporte privado (‘carsharing’, ‘carpooling’, gestión inteligente de flotas disociando propiedad del vehículo de su uso), como en el

transporte público (planes de oferta integral más competitivos, combinación de redes ferroviarias con redes de autobús y 'park&ride', etcétera).

El presente proyecto estudia la viabilidad tecnológica de una alternativa al modelo actual de transporte privado, que incrementaría la tasa de ocupación media de los vehículos, que actualmente se sitúa en 1,3 personas por vehículo en España.

1.1. Dynamic *Ridesharing*

El *dynamic ridesharing* consiste en emparejar automáticamente a una serie de usuarios, cada uno con un rol definido, en un plazo corto de tiempo. Los roles se dividen en dos, un conjunto de propietarios de vehículos, a partir de ahora denominados conductores y un conjunto de usuarios con una necesidad de transporte, denominados viajeros. Ambos conjuntos tienen sus propias rutas de transporte, pero están dispuestos a desviarse a cambio de compartir costes o reducir el impacto de su desplazamiento en el medioambiente. El hecho de compartir viaje disminuye la distancia global recorrida por el total de participantes.

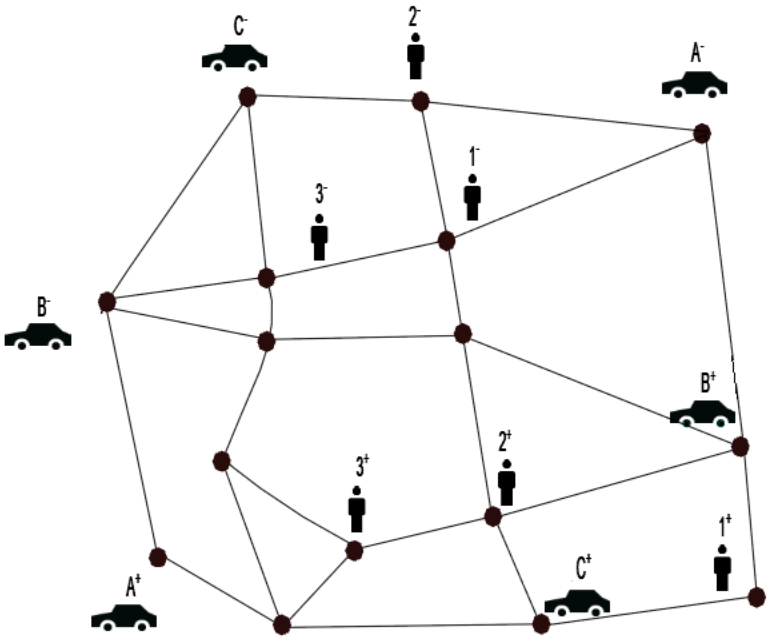


Figura 3.- Escenario de *ridesharing*

En la Figura 3 se muestra un escenario genérico de este modelo de transporte. En él se puede observar los dos conjuntos de participantes que se han comentado anteriormente. Los viajeros, representados por el símbolo del hombre, tienen unos puntos de origen y de destino (origen identificado por el signo + y destino por el signo -), al igual que el de los conductores. Cada uno de estos puntos son los nodos de una red

de transporte. Esta red de transporte se rige por unas reglas determinadas por el *ridesharing.*

El principal problema del *dynamic ridesharing* reside en el planteamiento del modelo de transporte y la optimización de este, en un tiempo razonable, con respecto a distintos parámetros como pueden ser el número de rutas que se emparejan o el total de distancia recorrida.

1.2. Objetivos

Como se ha dicho anteriormente, el problema principal a tratar en el proyecto es la viabilidad de los algoritmos de optimización para un modelo de transporte que cumpla con la funcionalidad del *ridesharing*, de tal manera que se consiga validar mediante simulaciones en distintos escenarios. Para llevar a cabo este objetivo principal es necesario distinguir distintos hitos u objetivos específicos:

- Investigación del problema y sus variantes, y estudio de distintas técnicas de optimización.

- Formulación de un modelo matemático adecuado a las necesidades de funcionamiento del *ridesharing*.

- Validación del modelo, comprobando que cumple las restricciones en entornos controlados.

- Búsqueda de un método de optimización para solucionar el problema en el menor tiempo posible con objeto de dar respuesta a las restricciones del *dynamic ridesharing*.

- Realización sistemática de simulaciones y evaluación de los datos obtenidos de estas. Análisis del efecto de las distintas variables de diseño sobre el resultado final. Evaluación de estos resultados bajo distintas métricas, como puede ser el porcentaje de distancia que se ahorra o el número de emparejamientos, ya que esto será muy importante para el siguiente objetivo.

- Análisis, a partir de los pasos anteriormente listados, de la viabilidad del *ridesharing* como modo alternativa al modelo tradicional de transporte privado.

1.3. Medios empleados

Para realizar este proyecto se ha utilizado la herramienta de cálculo MATLAB Versión 7.14.0.739 R2012a, que proporciona un entorno adecuado para el trabajo con magnitudes vectoriales y matriciales. Además contiene un conjunto de funciones estables y eficientes para la optimización, que se emplearán para la realización de simulaciones de entornos en los que se aplicará este modelo de transporte.

El equipo sobre el que se ha realizado el trabajo es un ordenador portátil LG E500 Intel(R) Core(TM)2 Duo CPU (@2.40GHz) y 3GB de RAM (6). Para la instalación de MATLAB Versión 7.8.0 R2012a el equipo mínimo es Intel Pentium 4 (1,4 y 1,5 GHz), 1GB de RAM y 1GB libres en el disco duro.

1.4. Marco de trabajo

El proyecto se ha desarrollado en el Centro de Automática y Robótica (CAR) de Consejo Superior de Investigaciones Científicas (CSIC) dentro del programa AUTOPIA.

El programa AUTOPIA ha venido trabajando en vehículos autónomos desde 1998 primero en la instrumentación con los sensores apropiados y luego en la automatización de los actuadores del coche (tanto desde el punto de vista de hardware como en los aspectos algorítmicos) y más recientemente en el desarrollo de maniobras cooperativas entre diferentes vehículos. El grupo tiene experiencia demostrada (un muy significativo índice de publicaciones en revistas de alto impacto, y participación en varios proyectos nacionales y europeos) en diseño de sistemas de control avanzados, localización, planificación de trayectorias, navegación y comunicaciones vehículo—vehículo y vehículo-infraestructura. AUTOPIA cuenta con una flota de 5 vehículos y un circuito de pruebas diseñado como un zona urbana con una combinación de tramos rectos y curvos, cruces a 90º y una rotonda. Además, un sistema de regulación de semáforos y redes de sensores RFID y ZigBee hacen de las instalaciones un excelente campo de pruebas para demostrar los más avanzados conceptos en el campo de los ITS.

1.5. Estructura de la memoria

Para facilitar la lectura de la memoria, a continuación se describe la estructura de esta. El trabajo está dividido en 7 capítulos:

- En el primero de ellos se ha descrito la motivación del proyecto y se ha realizado una breve introducción al problema, así como una descripción de los objetivos de este.

- El segundo capítulo recoge la información necesaria para describir en detalle el *ridesharing*, con los distintos escenarios que se podrían considerar y las diferencias con otros modos de transporte.

- En el tercer capítulo se formula el problema de forma matemática, describiendo tanto los parámetros de la función objetivo como los distintos tipos de restricciones del modelo que se ha desarrollado. Para ello se describen las alternativas que se han planteado para su solución.

- El cuarto capítulo describe los distintos métodos de optimización que se han empleado para solucionar el problema.

- En el quinto capítulo se realizan una serie de simulaciones de diferentes escenarios en los que se consideran variaciones en las distintas variables de diseño del modelo. Se realizará un análisis de sensibilidad y se analizarán los resultados de las simulaciones para obtener unas conclusiones.

- El sexto concluye el trabajo valorando los objetivos que se han cumplido, viendo la viabilidad del *ridesharing* como modo de transporte alternativo y marcando una serie de propuestas de trabajo futuras.

Capítulo 2

2. Dynamic *Ridesharing*

El *ridesharing* es un modo de transporte colectivo y responsable, que hace un uso óptimo de los recursos que tiene disponible. Consiste en la agrupación, en un mismo vehículo, de viajeros que comparten parcial o totalmente trayectos. El concepto deriva de otro anterior, el *car-sharing*. Históricamente se atribuye el nacimiento del concepto de *car-sharing* a Estados Unidos cuyo gobierno, durante la Segunda Guerra mundial, pidió a la ciudadanía que compartiera vehículo para ir al trabajo y de esa manera conservaran combustible para poder destinarlo a la guerra. Inglaterra también creó en la misma época los clubes de automóviles, en los que se compartía un vehículo alquilado entre varias personas. Lo más lógico parece creer que este concepto nace del sentido común, evitando derroches innecesarios y optimizando el uso de los recursos existentes.

También es conocido en el Reino Unido como *liftsharing* y *carsharing* (no confundir con los términos "*carsharing*" en el norte de América, y "*car-sharing*" o "clubes de automóviles" en el Reino Unido, que se refieren al alquiler compartido de automóviles durante un periodo corto de tiempo) se diferencia de los taxis y de otros transportes con fines de lucro (p.ej. empresas como Uber) en que sus motivaciones no son financieras. Cuando se realiza un pago en un viaje de *ridesharing*, se trata de cubrir parcialmente el costo del conductor. No se pretende que resulten beneficios económicos de ese trayecto.

Tabla 1.- Comparativa entre *ridesharing* y otros modos de transporte

	Transporte programado	**Transporte bajo demanda**	***Ridesharing***
Oferta			
Servicio	Transporte de punto A a B.	Transporte de puerta a puerta	Transporte de puerta a puerta
Capacidad	Flota de vehículos y conductores.	Flota de vehículos y conductores.	Vehículos propios del conductor
Ruta	Programadas	Bajo demanda	Bajo demanda/ Programada
Calidad del servicio	Trabajo regulado y tiempo de conducción restringido	Trabajo regulado y tiempo de conducción restringido	A preferencia del conductor
Dinámica de funcionamiento	Flota programada	Flota programada	Entrada dinámica de nuevos conductores
Demanda			
Beneficio	Precio del billete	Tarifa del taxi	Costes compartidos
Reserva	No necesaria	Desde semanas a varios minutos de antelación	Desde horas a varios minutos de antelación
Calidad del servicio	Exceso de tiempo de viaje	Reducción del tiempo de viaje	Reducción del tiempo de viaje, ahorro de costes
Dinámica de funcionamiento	Entrada de nuevas solicitudes	Entrada de nuevas solicitudes	Entrada de nuevas solicitudes

En la Tabla 1 se recoge el funcionamiento de algunos de los modos de transporte actuales. En ella se pueden ver las ventajas e inconvenientes que presenta el *ridesharing* respecto el resto de modos de transporte. Los modelos de transporte programado son muy rígidos, teniendo puntos de recogida fijos, flotas fijas, rutas programadas y horarios preestablecidos. Los modelos bajo demanda son más flexibles pero suelen ser más caros que los transportes programados. El *ridesharing* mejora respecto a los modelos de transporte programados, ya que el trayecto es de puerta a puerta, la flota se adapta a las necesidades, los horarios y rutas las pueden negociar los distintos participantes. También mejora respecto a los modelos de transporte bajo demanda ya que el coste del trayecto pasa de ser una tarifa, la cual incluye unos beneficios para el conductor, a ser una aportación a los gastos del viaje. De esta manera se compartirían los gastos sin generar más beneficio que el propio ahorro.

A continuación se muestra un ejemplo para aclarar el funcionamiento del *ridesharing*: El conductor, de color azul, anuncia que va a hacer su ruta y que está dispuesto a realizar un trayecto en el sistema de *dynamic ridesharing*. Un viajero, de color rojo, solicita al sistema una petición de recogida (Figura 4). La ruta del viajero, en este caso, acaba en una zona próxima a la del conductor.

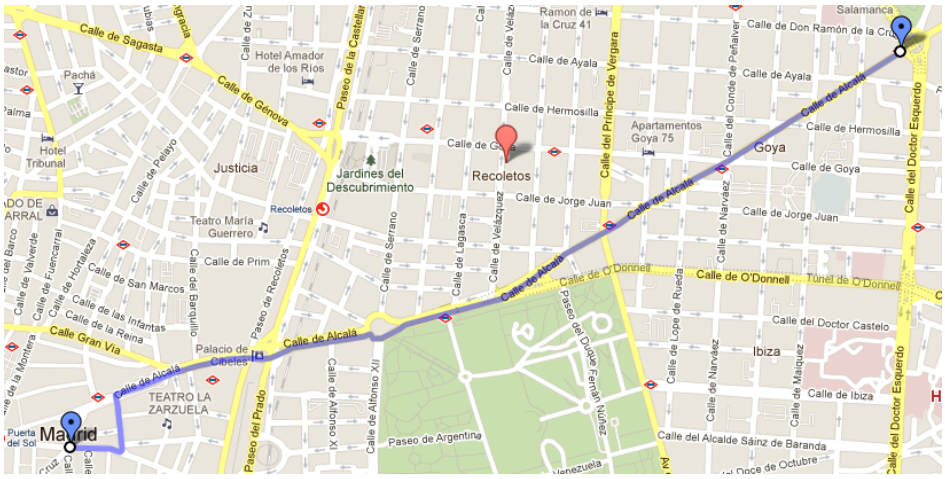


Figura 4.- Sistema *ridesharing* antes del emparejamiento.

El sistema de *dynamic ridesharing* busca cual de los conductores que actualmente están disponibles para realizar este tipo de servicio, en este caso el conductor azul. Calcula si la distancia extra que recorre el conductor si va a recoger al viajero, está dentro de un rango aceptable para el conductor y si compensa que vayan juntos, o es mejor que vayan cada uno por sus medios. En caso de cumplir estas condiciones, el sistema emparejará a ambos usuarios e indicará al conductor la nueva ruta para recoger al viajero (Figura 5).

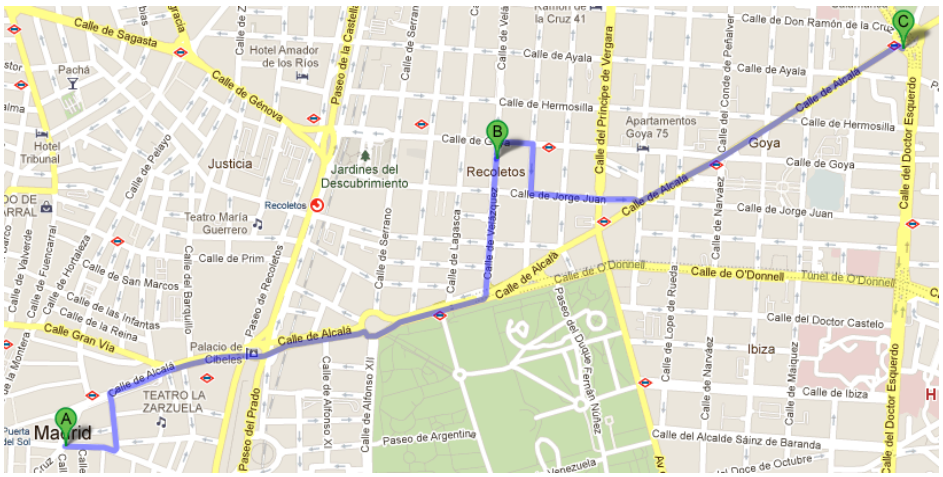


Figura 5.- Emparejamiento de rutas

Este modo de transporte no es aplicable únicamente en entornos urbanos, ya que la mayoría de desplazamientos diarios se producen de modo interurbano. Esto amplía el número de usuarios, haciendo que el *ridesharing* pueda ser una opción viable para disminuir atascos y contaminantes. La Figura 6 muestra un escenario típico en la comunidad de Madrid, en el que trabajadores de las afueras de Madrid acuden a sus puestos de trabajo situados en la zona noreste de la ciudad.

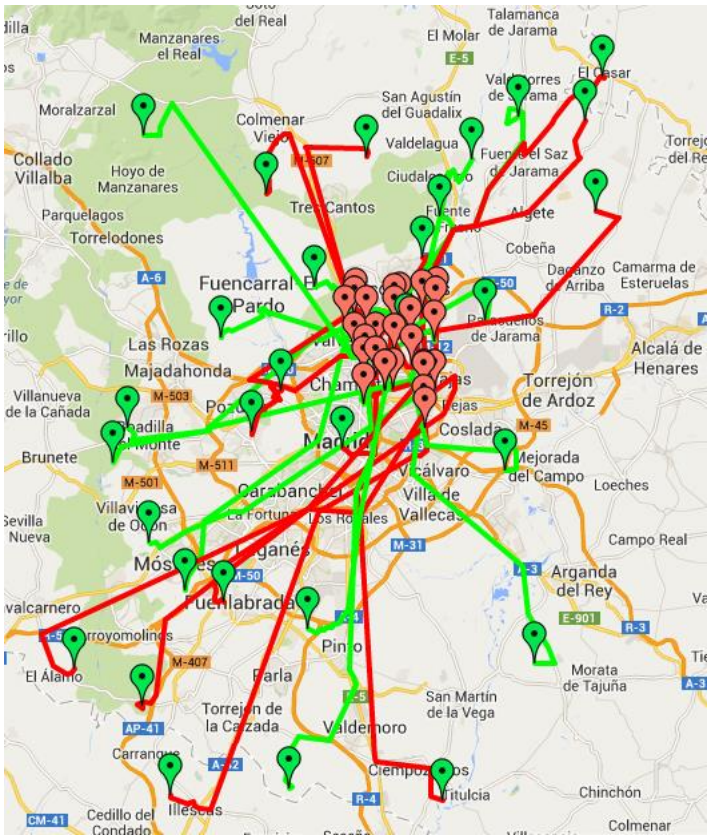


Figura 6.- Simulación del tránsito de trabajadores de las afueras de Madrid a la zona noreste de Madrid (en verde conductores, en rojo viajeros)

La mayoría de los coches pueden llevar al menos a cuatro pasajeros. En España el vehículo promedio tiene tasa de ocupación en la mayoría de sus viajes de 1,3 personas [3]. Estos asientos vacíos representan una fuente sin explotar de capacidad de transporte, y es por esto por lo que el *ridesharing* adquiere un gran interés.

El *ridesharing* reduce el número de automóviles necesarios, lo que produce numerosos beneficios para la sociedad. De ahí que la promulgación de políticas para aumentar *ridesharing* parezca la estrategia más eficaz para reducir el consumo de energía asociada a la conducción. Otros beneficios incluyen la reducción de emisiones, la descongestión del tráfico y demanda de infraestructura para aparcamiento. Sin embargo, la magnitud de tales beneficios no está clara ni es fácil de evaluar. El informe SMART 2020 [4] estima que el empleo de las tecnologías de la información y las comunicaciones (TIC) para optimizar la logística de transporte individual por carretera podría disminuir entre 70 y 190 millones de toneladas métricas de emisiones de dióxido de carbono. A parte de las emisiones de dióxido de carbono, se puede llegar a reducir a un 25 por ciento las emisiones del resto de gases de efecto invernadero (GEI) maximizando la ocupación un mismo vehículo (Figura 7).

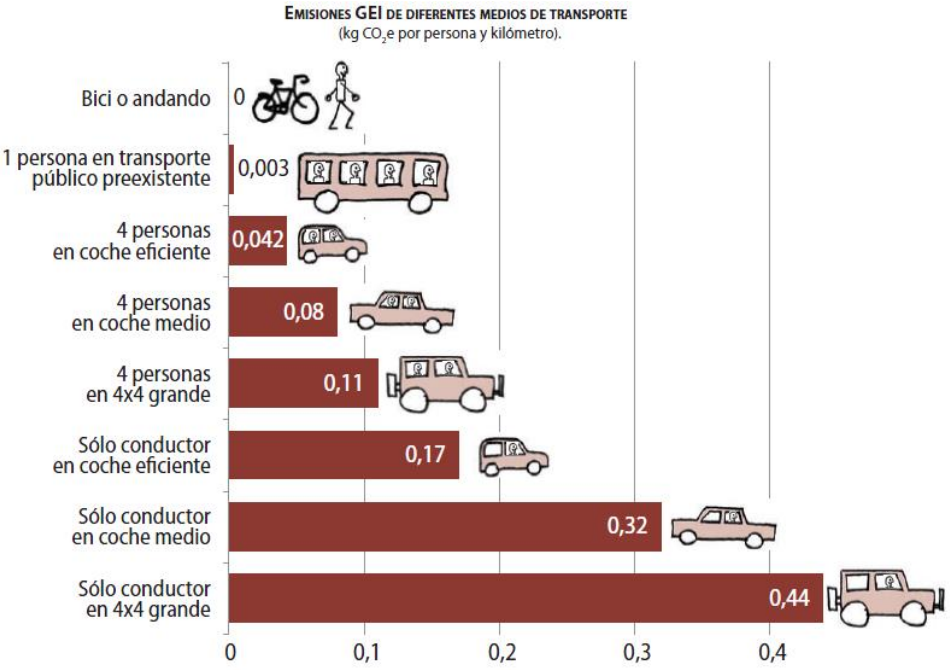


Figura 7.- Emisiones de GEI de diferentes medios de transporte

Se suele referir a menudo al hecho de compartir coche como el "modo invisible" de transporte. Esto es porque es difícil de observar y estudiar. Hay poca documentación de la historia del *ridesharing* o de servicios de coche compartido y pocos datos cuantitativos, simplemente porque estos trayectos son difíciles de registrar y contar.

A nivel individual, los beneficios son más tangibles. Los participantes en estos tipos de servicios ahorran costes debido a que los gastos asociados al viaje son compartidos. Pueden ahorrar tiempo de viaje por el empleo de carriles para vehículos de alta ocupación (Carriles BUS-VAO). Además, en algunos países se tiene acceso a estacionamiento preferencial y otro tipo de incentivos.

La compañía Amovens, compañía para compartir vehículo en trayectos con origen y destino común, aporta un estudio [5] en el que se describen los beneficios de compartir vehículo. Estima que se pierde en atascos al ir a trabajar alrededor de 7 días al año, cuyo equivalente económico es de 538 euros. Para un usuario que comparte vehículo para ir a trabajar a una distancia de 20 km de su domicilio, el ahorro anual es de, aproximadamente, 1500 euros. Estos 1500 euros equivalen aproximadamente a los gastos domésticos de una vivienda de dos personas al año. También beneficia a las empresas que lo fomentan, ya que esto provoca que haya más plazas de aparcamiento libres. Para viajes largos, compartir coche permite reducir los costes de viaje de larga distancia entre un 50% y un 80% con respecto a conducir solo. Para un viaje desde Madrid a Valencia la tabla a continuación recoge la diferencia entre distintos métodos de transporte. En la Tabla 2.- Resumen de costes y emisiones de distintos modos de transporteTabla

2 Se puede observar que el coste y emisiones son menores conforme se incrementa la ocupación, en lo que a automóviles se refiere.

Tabla 2.- Resumen de costes y emisiones de distintos modos de transporte

Modo de transporte	Coste (€)	Emisiones (kg/persona)
Vehículo de gama media con un ocupante	54	60
Autobús	28	18
Tren	60	10
Avión	115	43
Vehículo medio ocupado por 3 personas	15	15

A pesar de sus muchas ventajas, existen barreras de comportamiento frente al incremento de uso del vehículo compartido. Las impresiones que puede producir el usuario al compartir vehículo pueden ser atractivas por el hecho de compartir gastos, pero los usuarios pueden estar poco dispuestos a sacrificar la flexibilidad y la comodidad de los automóviles privados. La seguridad personal es una preocupación cuando se comparte un trayecto con desconocidos, aunque se trata de un riesgo percibido.

2.1. Objetivos del *ridesharing*

Una vez conocido el concepto de *ridesharing*, se pueden definir los objetivos de este modo de transporte:

- Minimizar la distancia total recorrida por la flota de vehículos: Esta distancia es la suma de las distancias recorridas por los participantes viajando a su destino, ya sea por su cuenta o empleando el *ridesharing.* Este objetivo es importante desde el punto de vista social ya que reduce la contaminación (emisiones) y la congestión de nuestras ciudades. Este objetivo está directamente relacionado con minimizar el coste total del viaje, siendo esto de gran importancia ya que es la motivación de los participantes para realizar este tipo de prácticas.

- Minimizar el tiempo total de viaje: Este tiempo es el que transcurre desde un origen a un destino. No consiste en reducir el tiempo de un conductor, ya que este sacrificará un tiempo extra que no tendría que emplear si fuera directamente a su destino. Consiste en reducir el tiempo en el que cada usuario está haciendo uso de la infraestructura de transporte. Si se comparte viaje, temporalmente será mayor para el conductor pero la infraestructura pasa de tener dos trayectos independientes a tener uno algo más largo, pero siempre

menor que la suma de los dos independientes. Desde el punto de vista social, esto reduce también las emisiones.

- Maximizar el número de participantes: Este objetivo maximiza el número de conductores y viajeros que se ven beneficiados por este sistema. Este objetivo puede ser beneficioso para el proveedor de *ridesharing* ya que cuando mayor sea el número de usuarios, mayor serán sus beneficios. El indicador de emparejamiento de rutas será definitivo a la hora de optar por un servicio y otro de *ridesharing*.

2.2. Estructura del sistema de *ridesharing*

La estructura software y hardware del sistema quedan fuera de estudio en este trabajo, pero es necesaria una idea básica de la topología del sistema para poder entender su funcionamiento. El funcionamiento del sistema se basa en una red centralizada. Consiste en un servidor centralizado que almacena y procesa las peticiones y ofertas de servicio de los conductores y viajeros. Los participantes usan sus dispositivos móviles (tablets, smartphones, …) para comunicarse con el servidor mediante internet. Los conductores declararán su intención de participar en el *ridesharing*, almacenando el servidor las rutas del conductor. Cada ruta almacenada consiste en las localizaciones de origen y de destino y otra información, como el máximo desvío que está dispuesto a realizar el conductor. Del mismo modo, los viajeros almacenan sus solicitudes de viaje con las mismas características que el conductor, su localización de origen y de destino. El servidor será el encargado de realizar el emparejamiento de conductores y viajeros, enviando a los conductores la ruta a seguir. La Figura 8 muestra el sistema descrito anteriormente. Se añade el uso de redes sociales para que emparejar a los viajeros como otra fuente de posibles usuarios, ya que para que el sistema muestre un beneficio social considerable se necesita llegar a una masa crítica de usuarios.

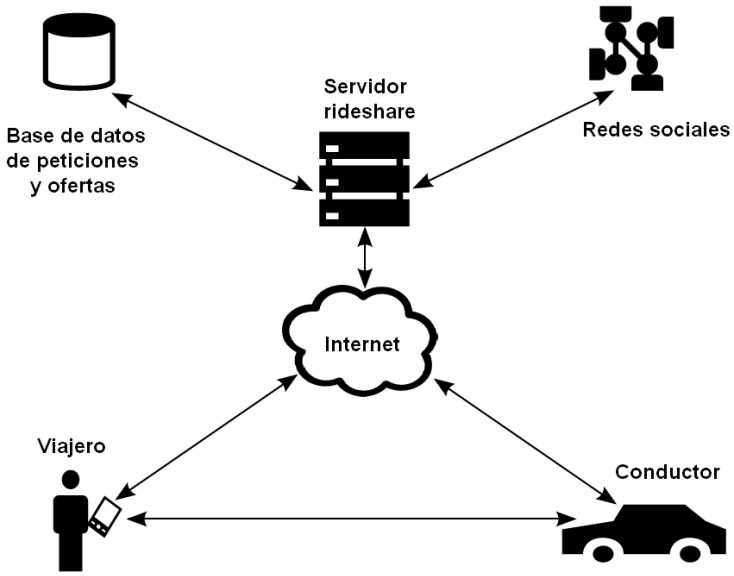


Figura 8.- Topología de un sistema de *ridesharing* genérico

2.3. Servicios existentes

La etiqueta dinámica dentro del *dynamic ridesharing* hace alusión a un sistema que gestiona peticiones y realiza emparejamientos en un periodo corto de tiempo o incluso en marcha. Empresas creadas reciente como Carticipate, EnergeticX, Avego o Piggyback proveen este tipo de servicios. Cuentan con aplicaciones para móviles con sistemas de localización GPS. Para evitar problemas de desconfianza a la hora de compartir viaje con un extraño, estos servicios tienen sistemas de evaluación de usuarios (PickupPal) y cuentan con herramientas conectadas con redes sociales como Facebook (empresas como GoLoco y Zimride).

En España existen ejemplos similares como pueden ser Blablacar, Amovens o Uber. Estos se basan en el hecho de compartir vehículo, y los costes asociados a su desplazamiento, pero no comparten la filosofía del *dynamic ridesharing*. Compañías como Blablacar necesitan de una antelación del orden de horas para poder establecer un viaje, por lo que la cualidad dinámica del *dynamic ridesharing*. Uber si que funciona de forma dinámica, pero la tarifa que se establece entre conductor y viajero la fija el viajero, provocando esto que haya casos en los que no se comparten los gastos del viaje, sino que los conductores reciben unos beneficios [noticia la informacion]. Esto diferencia el funcionamiento de Uber del que se trata en este trabajo, en el que el uso del *ridesharing* no reside en obtener beneficios sino en compartir los costes asociados al viaje. En esta versión de *ridesharing* los costes asociados al viaje los calcularía el servidor de *ridesharing* mediante información relativa al vehículo (coste del seguro, consumos medios en ciudad y en recorridos interurbanos, etc).

Capítulo 3

3. Descripción del problema

En este capítulo se describirá el núcleo del problema a resolver. Este problema puede presentar ciertas variantes que quedan fuera del estudio de este trabajo, pero es necesario conocer. Por último se formulará, de forma descriptiva, en comportamiento del sistema y las condiciones que debe cumplir para tener la funcionalidad descrita en el capítulo anterior.

3.1. *Ridematching*

Para resolver el problema del *ridesharing* primero se debe resolver otro problema, el *ridematching* o emparejamiento de viaje. El *ridematching* es el método de emparejamiento automático entre conductores y pasajeros que requiere del menor esfuerzo por parte de sus participantes. Dado un grupo de participantes, diferenciados entre conductores y viajeros, un conductor podría elegir entre compartir su viaje con un solo pasajero o estar dispuesto a recoger a varios pasajeros durante su viaje. Del mismo modo, un pasajero podría estar interesado en hacer su viaje con un solo conductor o puede permitir al sistema emparejarlo con más de un conductor para alcanzar su destino, emparejándose a distintas horas con distintos conductores. Por lo tanto se pueden definir cuatro variantes de *ridematching* (emparejamiento):

- Un conductor con un pasajero (SDSR – Single Driver Single Rider).
- Un conductor con múltiples pasajeros (SDMR – Single Driver Multiple Rider).
- Múltiples conductores con un pasajero (MDSR – Multiple Driver Single Rider).
- Múltiples conductores con múltiples pasajeros.

En la Tabla 3 se muestran las distintas funciones que se deben tener en cuenta en las variantes de *ridematching*, así como sus principales problemas a resolver:

Tabla 3.- Requisitos de los distintos modos de *ridesharing*

	Un pasajero	**Múltiples pasajeros**
Un conductor	Emparejamiento entre pares de conductor/pasajero.	Emparejamiento y creación de rutas para recoger y entregar pasajeros.
Múltiples conductores	Emparejamiento y creación de rutas para el intercambio entre conductores.	Emparejamiento y creación de rutas para recoger y entregar pasajeros y para el intercambio de pasajeros entre conductores.

3.1.1. Variante Single Driver and Single Rider (SDSR)

La primera del problema es el emparejamiento de un conductor con un pasajero (SDSR). En este caso, un pasajero puede emparejarse como máximo con un conductor y un conductor puede emparejarse como máximo con un pasajero. El principal problema en esta variante es encontrar el mejor emparejamiento entre pares de conductor/pasajero considerando una función objetivo. Si se empareja un conductor con un pasajero, se crea una ruta directa. El conductor sale de su lugar de origen, recoge al pasajero y lo lleva a su destino. Después el conductor viaja a su destino, como se muestra en la Figura 9.

Figura 9.- Modelo Single Driver - Single Rider (SDSR)

Los círculos representan los nodos. Los orígenes se denotan por el símbolo + y los destinos por el símbolo -. El color gris corresponde con los nodos del conductor y el blanco con el de los viajeros.

3.1.2. Variante Single Driver and Multiple Riders (SDMR)

En la variante de un conductor y múltiples pasajeros (SDMR), el conductor puede emparejarse con más de un pasajero mientras que el pasajero sólo se puede emparejar con un único conductor. Esta variante es más compleja de resolver que la anterior. Esta vez consta de dos problemas que son el emparejamiento de los conductores y pasajeros y la creación de las rutas de los conductores. En este caso se deberá encontrar la mejor combinación de emparejamientos entre conductores y pasajeros teniendo en cuenta alguna función objetivo de tal modo que el conductor se puede emparejar con múltiples pasajeros y cada pasajero se debe emparejar como máximo con un conductor. Además se debe crear la ruta de los conductores en la que se describe el orden en el que se vista los puntos de recogida y entrega de cada pasajero. No es necesario visitar el origen y el destino del pasajero de manera consecutiva (Figura 10), lo que hace más complicado el problema.

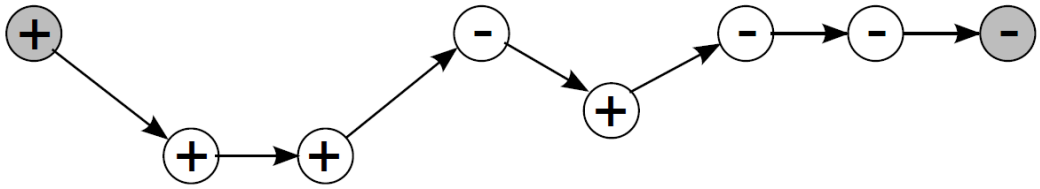


Figura 10.- Modelo Single Driver - Multiple Rider (SDMR)

3.1.3. Variante Multiple Drivers and Single Rider (MDSR)

En la variante de múltiples conductores y un pasajero se permite al pasajero compartir su viaje con varios conductores a diferentes horas mientras que los conductores solo pueden emparejarse con un único pasajero. Los problemas de esta variante son el emparejamiento y la creación de ruta de los pasajeros. El emparejamiento debería encontrar las mejores parejas de conductores y pasajeros teniendo en cuenta una función objetivo en la que cada pasajero se puede emparejar con múltiples conductores, y cada conductor se puede emparejar como máximo con un pasajero. La creación de las rutas de los conductores es fácil en este caso. Como en el caso de SDSR, el conductor viaja desde su origen al origen del pasajero para llevarlo a su destino. Tras esto, el conductor viaja a su destino. La tarea compleja es la creación de las rutas de los pasajeros. Si un pasajero se empareja con múltiples conductores, su ruta debe ser tal que la conectividad espacial y temporal se mantenga como se muestra en la Figura 11. El conductor debe entregar al pasajero en el sitio donde el siguiente conductor le recogerá. Además, el conductor debe entregar al pasajero antes de la hora a la que el siguiente conductor le recogerá en el punto de la ruta del pasajero

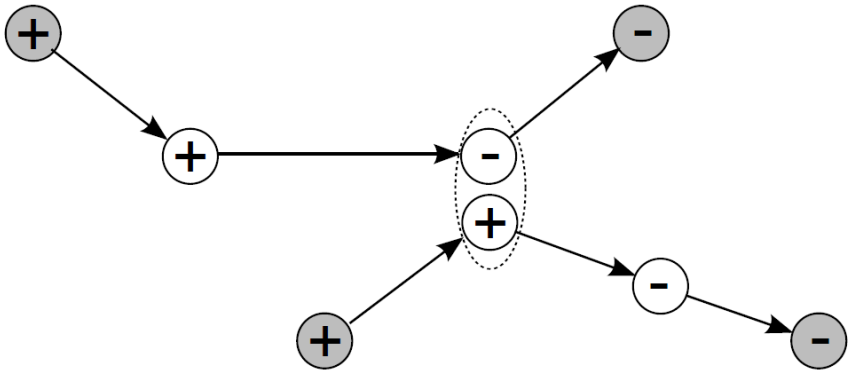


Figura 11.- Modelo Multiple Driver - Single Rider (MDSR)

El primer conductor recoge al viajero y lo entrega en un destino intermedio. En este destino intermedio o punto de transbordo, el viajero espera a otro conductor (elipse discontinua) que complete la ruta del viajero.

3.1.4. Variable Multiple Drivers and Multiple Riders (MDMR)

En la variante de múltiples conductores y múltiples pasajeros un conductor puede ser emparejado con múltiples pasajeros, y un pasajero puede ser emparejado con múltiples conductores. Esta variante combina las complejidades del SDMR y el MDSR. Los problemas de esta variante son el emparejamiento de conductores y pasajeros y la creación de las rutas de ambos. La ruta de un conductor tiene que pasar por los orígenes y destinos de los pasajeros y la ruta de un pasajero tiene que pasar por diferentes conductores como se muestra en la Figura 12

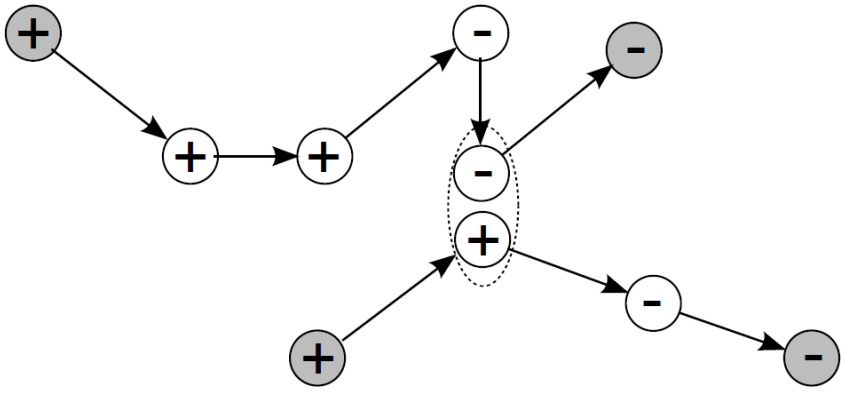


Figura 12.- Modelo Multiple Drivers - Multiple Riders (MDMR)

El primer vehículo recoge a dos viajeros distintos. Deja a uno de ellos en su destino y deja al otro en un punto de transbordo. Tras esto, el conductor completa su ruta y el viajero espera a que otro conductor distinto le recoja y complete su ruta.

3.1.5. Ventanas temporales.

Previamente se han descrito las distintas variantes a la hora de realizar los emparejamientos. El *dynamic ridesharing* puede considerar un requisito que puede hacer que las variantes de emparejamientos sean aún más difíciles de resolver, pero hacen que el concepto de *ridesharing* sea más práctico. Este requisito es la consideración de ventanas temporales que limitan la hora de salida y de llegada, como se explicará a continuación.

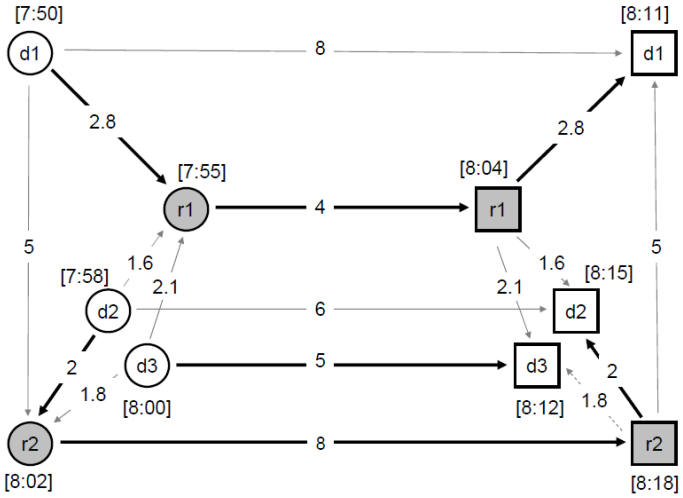


Figura 13.- Modelo con ventanas temporales

La Figura 13 muestra un ejemplo de ventanas temporales en las que ambos, conductores y viajeros, tienen una restricción temporal que limita la salida y llegada de estos.

Los participantes deben especificar las ventanas temporales en las que están dispuestos a compartir sus viajes. Las ventanas temporales especifican la hora de salida más temprana y más tardía, así como la hora de llegada más temprana y más tardía. Por ejemplo, un participante especifica que como muy pronto quiere salir a las 09:30 y como muy tarde a las 10:00 para llegar a su destino a tiempo. Un viaje antes de la hora más temprana no sería adecuado ya que se ha especificado que se desea salir a partir de las 09:30. Un viaje después de la hora más tardía es inaceptable, ya que puede no llegar a su destino a tiempo. Los participante también pueden especificar cuánto tiempo/distancia extra están dispuestos a viajar si participan en el *ridesharing*.

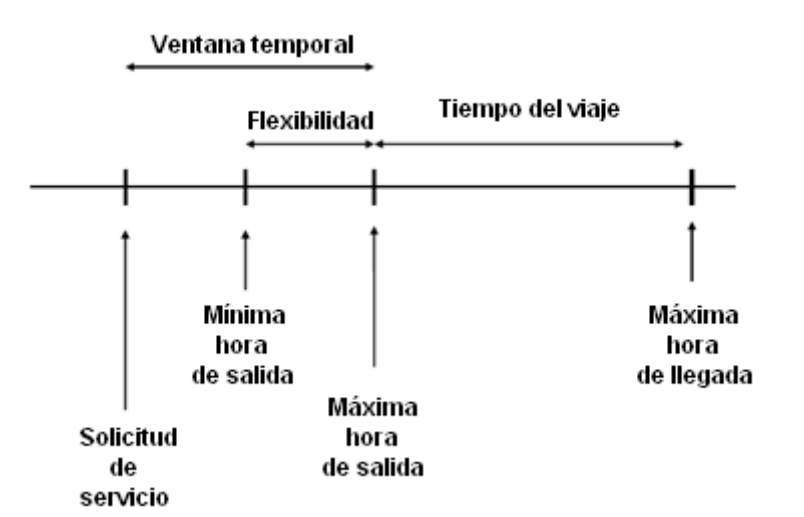


Figura 14.- Ventanas temporales

En la Figura 14 se observan los distintos intervalos temporales que se deben tener en cuenta. El primer intervalo es la ventana temporal para el emparejamiento. El tamaño de este intervalo se define por: el tiempo que pasa entre que se anuncia la solicitud de servicio y se realiza el emparejamiento, y la flexibilidad que presente el participante a la hora de salir de su origen. Es importante que el método de emparejamiento sea rápido para dar un mejor servicio, permitiendo así dar servicio aun cuando se solicita con poca antelación. El segundo intervalo es el debido al desplazamiento en el propio viaje, que debe ser el menor posible ya que se pretende optimizar las rutas para disminuir en lo posible el tiempo o distancia de viaje.

Incluir las ventanas temporales en el problema lo hace más complicado de resolver, especialmente si el tiempo total de viaje se considera en la función objetivo (variable a minimizar) y si el tiempo de viaje de los participantes está restringido. Adelantar o retrasar la salida de un participante puede incrementar las posibilidades de aumentar el número de emparejamientos o disminuir la distancia/tiempo de viaje.

En estudios relacionados con la solución de este tipo de problemas se realiza un análisis del estado actual de las distintas variantes, así como de las soluciones que se han planteado [6]. De este estudio se deduce que a pesar de que el problema del emparejamiento es el concepto central del *ridesharing*, aún no se ha invertido suficiente esfuerzo en solucionarlo. Estos problemas se vuelven exponencialmente complejos, lo que hace necesario desarrollar algoritmos específicos de optimización para solucionarlos.

La Tabla 4 muestra un resumen de los requisitos que se cumplen para las distintas variantes. En ella se hace referencia a los requisitos básicos, que en este caso son los requisitos de emparejamiento y las restricciones temporales.

Tabla 4.- Métodos resueltos de forma optima

		Un pasajero	Múltiples pasajeros
Un conductor	• Requisitos básicos • Ventanas temporales	Totalmente Parcialmente	Parcialmente No cumple
Múltiples conductores	• Requisitos básicos • Ventanas temporales	Parcialmente No cumple	No cumple No cumple

Este trabajo se centra en analizar la variante de un conductor y un viajero (SDSR). Las variantes en las que se implican múltiples conductores o viajeros, o las que trabajan con restricciones temporales convierten al problema en MINLP, problema no lineal de enteros mixtos. Los métodos para solucionar este tipo de problemas son capaces de encontrar una solución que cumpla con las restricciones y la función objetivo, pero no se basan en métodos exactos en los que se garantiza la solución óptima, debido a la complejidad de estos. Es interesante conocer la solución óptima para evaluar el modelo.

MINLP (Mixed Integer Nonlinear Programming) hace referencia a problemas matemáticos con variables continuas y discretas y no linealidades tanto en su función objetivo como en las restricciones.

Los problemas MINLP son difíciles de resolver porque combinan todas las dificultades de sus subclases: la naturaleza combinatoria de los problemas de enteros mixtos (MIP, de sus siglas en inglés Mixed Integer Problem) y la dificultad de resolver problemas no lineales (NLP, de sus siglas en inglés Nonlinear Problem). Existen métodos determinísticos [7] para resolver este tipo de problemas, pero necesitan tiempo suficiente para ello y su coste computacional es inasumible cuando e número de agentes comienza a crecer. Otra manera de resolver estos problemas es con el uso de los métodos heurísticos [8], que no garantizan una solución óptima y tienen un tiempo de convergencia muy variable.

3.2. Descripción del modelo

Los problemas de transporte son afrontados día a día por todo el mundo. Desde los más simples, como organizar la ruta hacia el lugar de trabajo, hasta los más complejos, como pueden ser la gestión de grandes flotas de vehículos por parte de compañías logísticas, ambos necesitan un buen modelo que defina las características del problema de la manera más realista posible [9]. El problema del *ridesharing* pertenece

a esta familia de problemas. Estos consisten en un conjunto de rutas para los vehículos implicados, que se basan en un origen y un destino determinados por el conductor y una serie de posibles puntos intermedios que son determinados por las peticiones de servicio de los viajeros. Para el problema a tratar, estos puntos intermedios pueden ser visitados únicamente por un conductor.

Para la resolución del problema se debe recrear el comportamiento del sistema de transporte. Para ello se deben definir las reglas de funcionamiento de éste, así como el objetivo que debe cumplir. Esto implica construir un modelo matemático que se adapte al funcionamiento descrito en capítulos anteriores para poder realizar simulaciones que validen su comportamiento. A continuación se abordarán, de forma descriptiva, las distintas restricciones que deberá cumplir el modelo así como las variables de diseño que constituirán el modelo.

En la Figura 15 se muestra un problema clásico de transporte, el problema del viajante o TSP (de sus siglas en inglés, Travelling Salesman Problem), para ilustrar los distintos elementos de los que se compone un problema de transporte.

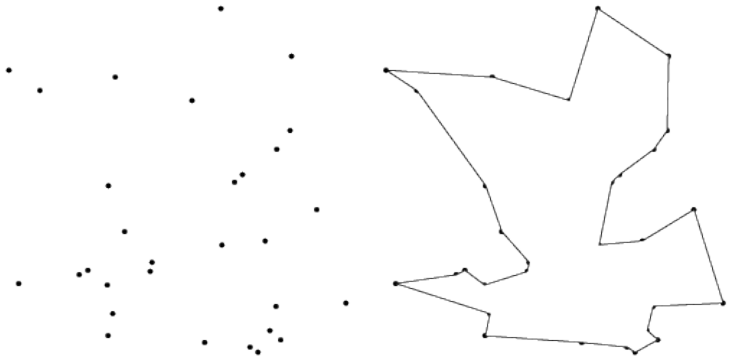


Figura 15.- Ejemplo TSP simple

En la Figura 15 se muestra en la parte izquierda una serie de puntos, o nodos. Estos nodos son las posiciones de origen o destino. En la parte derecha se puede observar que estos nodos han sido unidos por una línea. Esta línea se denomina ruta y establece la conexión entre un nodo de origen y uno de destino. La ruta total de este problema TSR sería el conjunto de líneas. A priori puede parecer trivial la resolución de este problema, pero la resolución óptima no es siempre sencilla y se vuelve exponencialmente compleja conforme aumenta el número de nodos.

3.2.1. Descripción de los elementos del modelo

A continuación se realizará una descripción de cada uno de los elementos de los que se compone un modelo de transporte.

- **Variables del problema:**

El primer objeto de estudio serán las variables de diseño del problema. Estas variables son cada una de las posibles rutas que se consideran en el problema. Una ruta se define como el trayecto entre dos nodos, como se ha descrito en el apartado anterior. Este conjunto de rutas será la solución del problema.

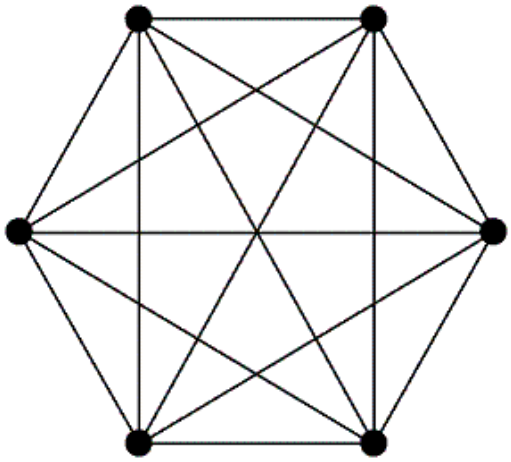


Figura 16.- Ejemplo de nodos y rutas

En la Figura 16 se muestran todas las posibles rutas que existen entre los distintos nodos del sistema. Estas variables son binarias lo que implica que esa ruta tiene dos posibles estados: la ruta se realiza, o no se realiza. Cada ruta tiene asociada una distancia.

Para el problema, las rutas tienen distintos identificadores que indicarán el punto de origen, el punto de destino y el vehículo en el que se está viajando. Este aspecto se describirá con mayor detalle en la formulación matemática del problema.

- **Función objetivo:**

Como se ha descrito en el capítulo anterior, el objetivo de este modelo de transporte es el de minimizar la distancia total recorrida, por lo que la función objetivo de este modelo matemático debe reflejar este comportamiento. Esto implica que la solución óptima tiene que configurar una serie de rutas de tal modo que la suma de las distancias de estas rutas sea la mínima posible, siempre que se cumplan las restricciones.

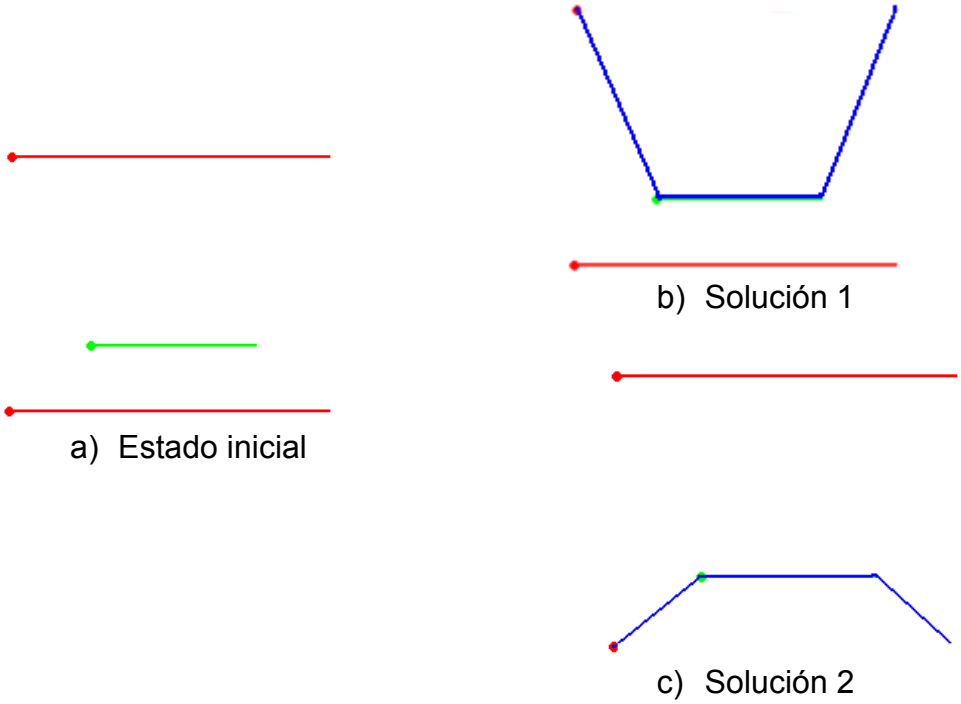


Figura 17.- Escenario con dos conductores (rojo) y un viajero (verde).

En la Figura 17 se muestra un escenario de *ridesharing* (a) en el que participan dos conductores (líneas de color rojo) que van desde sus nodos de origen (punto grueso rojo) hasta sus nodos finales (final de la línea) y un viajero que va desde su nodo de origen (punto grueso verde) hasta su nodo final (final de la línea verde). Existen dos posibles soluciones en las que un conductor recoja a un viajero, pero únicamente una es la solución óptima del problema. La solución 1 (b) muestra que el conductor de la parte superior recoge al viajero, pero se puede observar a simple vista que la ruta que recorre éste es mayor que la que recorre el vehículo de la parte inferior en la solución 2 (c), por lo que esta última será la solución óptima.

- **Restricciones:**

Las restricciones determinan que rutas se pueden hacer, establecen las incompatibilidades entre rutas que son parte del comportamiento del sistema.

La primera restricción establece la primera ruta del conductor. Esta ruta debe de ser única, evitando así que el vehículo vaya a la vez a dos destinos distintos. Los destinos de esta primera ruta pueden ser el destino de la ruta del conductor, realizando la ruta directa, o cualquiera de los orígenes de los viajeros.

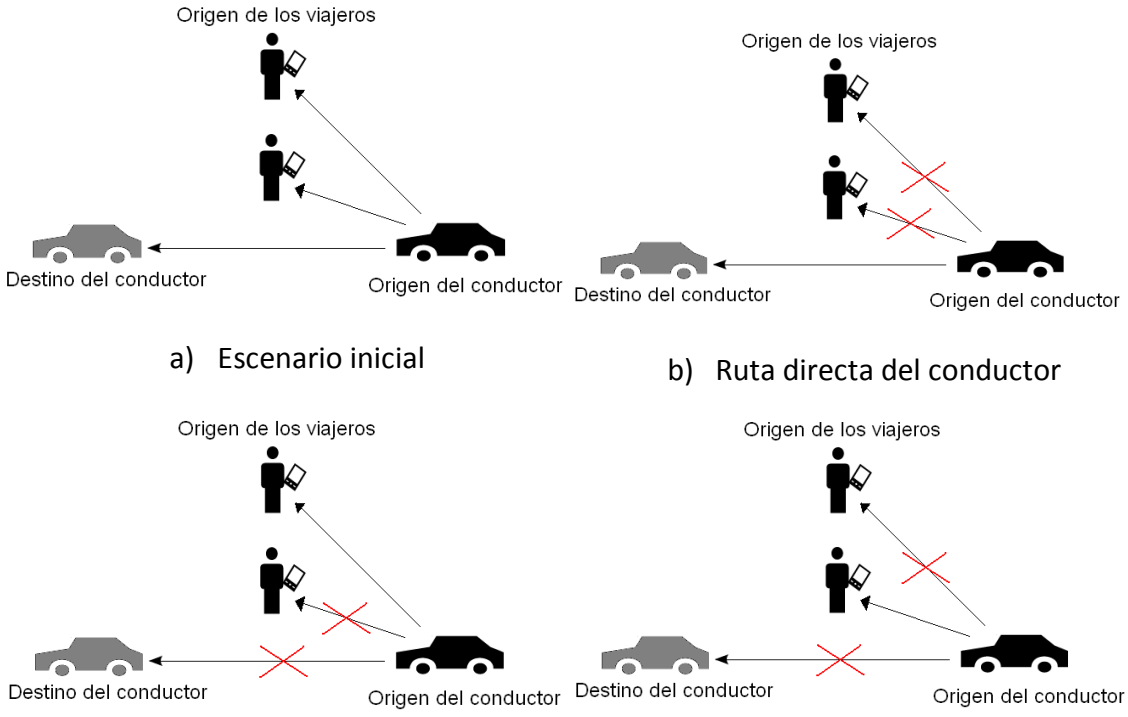
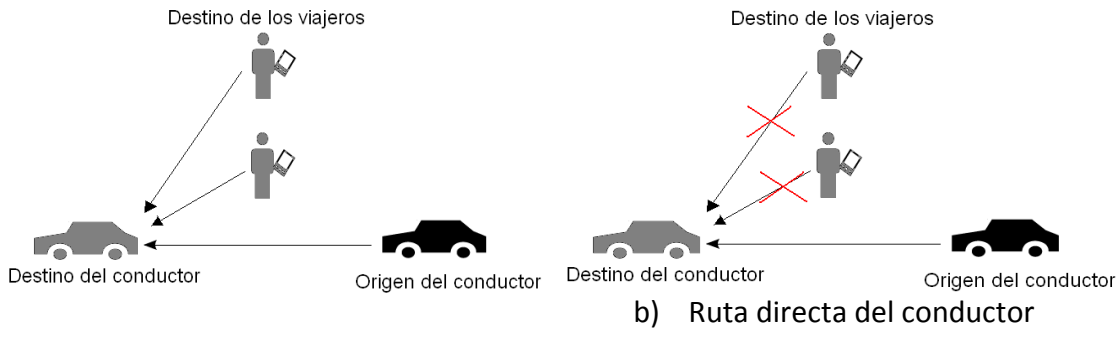
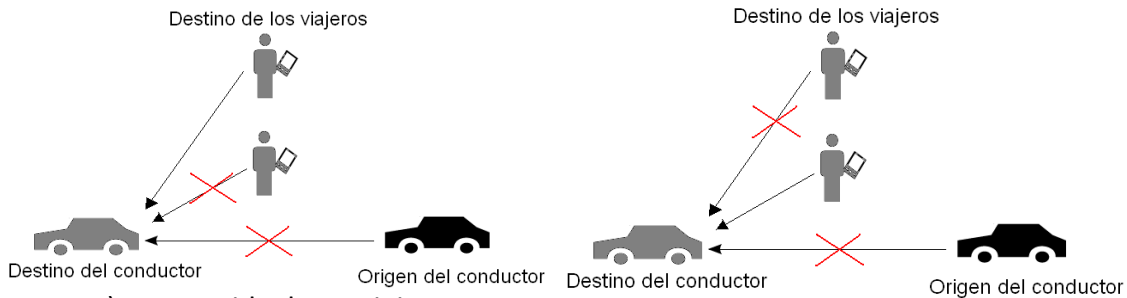


Figura 18.- Representación gráfica de la primera restricción

La figura anterior muestra un ejemplo de aplicación de la primera restricción. La figura a) muestra el total de posibilidades de ruta inicial para el vehículo. En la figura b) el vehículo viaja directamente de su origen a su destino. En las figuras c) y d) realiza una recogida de viajeros distintos.

La segunda restricción establece la última ruta del conductor. Esta ruta, al igual que la establecida por la restricción anterior, debe de ser única, evitando así que el vehículo venga a la vez de dos orígenes distintos. Los orígenes de esta última ruta pueden ser el origen de la ruta del conductor, realizando la ruta directa, o cualquiera de los destinos de los viajeros (Figura 19).





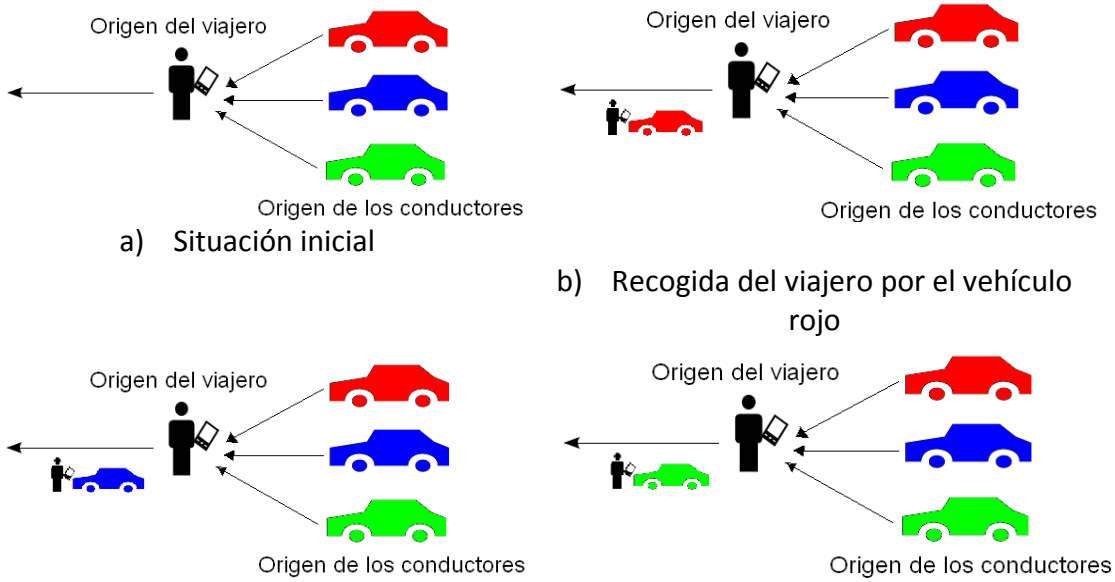
c) Recogida de un viajero

d) Recogida del otro viajero

Figura 19.- Representación gráfica de la segunda restricción

La figura anterior muestra un ejemplo de aplicación de la segunda restricción. La figura a) muestra el total de posibilidades de ruta final para el vehículo. En la figura b) el vehículo viaja directamente de su origen a su destino. En las figuras c) y d) realiza una entrega de viajeros distintos.

La tercera restricción asegura que un viajero se empareja para una ruta como máximo con un único conductor.



a) Situación inicial

b) Recogida del viajero por el vehículo rojo

c) Recogida del viajero por el vehículo azul

d) Recogida del viajero por el vehículo verde

Figura 20.- Representación gráfica de la tercera restricción

La Figura 20 muestra un ejemplo de aplicación de la tercera restricción. La figura a) muestra el total de posibilidades de emparejamiento para el viajero en los distintos vehículos del sistema. En las figuras b), c) y d) el viajero viaja con distintos vehículos, estableciendo la ruta únicamente con uno de ellos.

La cuarta restricción asegura la continuidad de una ruta. Esto significa que el conductor que recoge al viajero en su origen será el mismo que lo entregue en su

destino. Para ello la ruta que trasporta al viajero de su origen a su destino tiene que ser en el mismo vehículo que recoge al viajero en su origen y el que lo entrega en su destino.

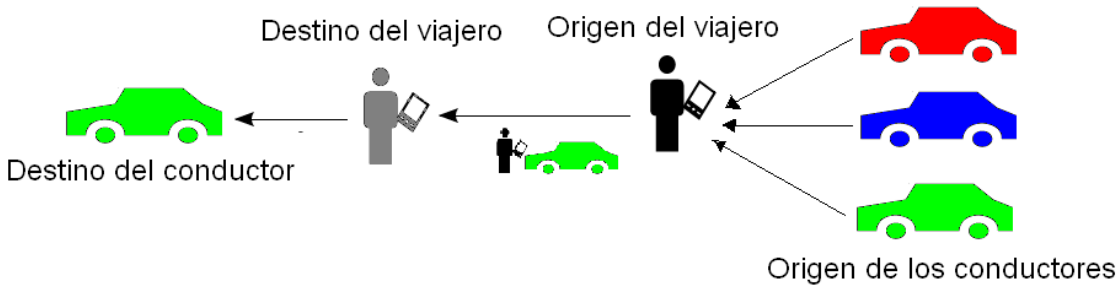
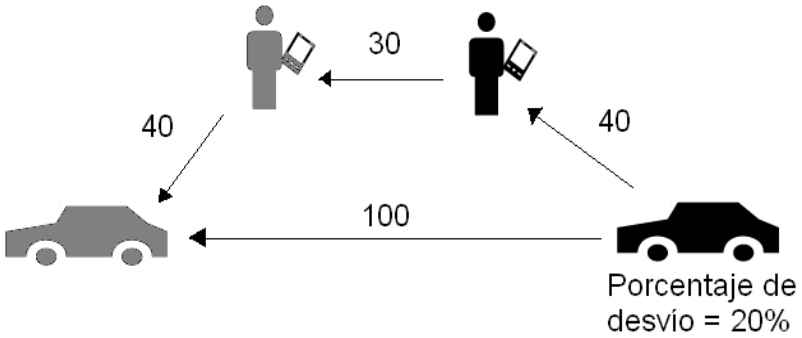


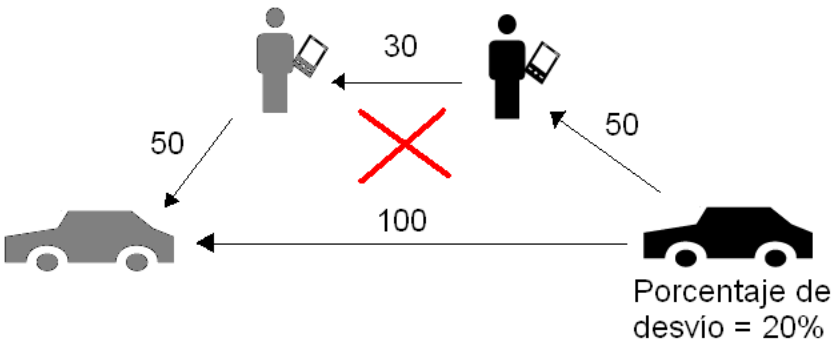
Figura 21.- Represtación gráfica de la cuarta restricción

La Figura 21 muestra el emparejamiento del conductor verde con un viajero. Estos dos comparten la ruta del viajero, hasta su punto de destino. Una vez realiza la entrega, el conductor viaja a su destino. El conductor que recoge al viajero es el mismo que viaja en la ruta del viajero y, tras la entrega, viaja a su destino.

La quinta restricción permite a un conductor establecer una distancia máxima que está dispuesto a desviarse para recoger a un viajero.



a) Ruta con distancia de desvío válida.



b) Ruta con distancia de desvío inválida.

Figura 22.- Representación de la quinta restricción

La Figura 22 muestra dos ejemplos de aplicación de esta restricción. En la figura a) un vehículo está dispuesto a desviarse un 20% de su propia ruta. Para recoger al viajero, entregarlo en su destino y finalizar la ruta del propio vehículo la ruta que recorre es de 110. Esto es menor que el desvío del 20% de la ruta del vehículo (la máxima ruta recorrida por el vehículo sería de 120) por lo que se realizará el emparejamiento. La ruta b) no cumple con esta distancia de desvío ya que la ruta, en caso de realizarse el emparejamiento, sería de 130.

Capítulo 4

4. Formalización del problema

Cuando se habla de minimizar la distancia de las rutas y maximizar el número de emparejamientos, se está hablando de optimización. El concepto de optimización se basa en encontrar la solución de menor coste o mayor rendimiento cumpliendo unas restricciones dadas, mediante la maximización de los factores deseados y minimización los no deseados.

En este capítulo se describen conceptos básicos de optimización y programación lineal necesarios para comprender la notación y formalización del problema de forma matemática. También se describen una serie de algoritmos de optimización que se emplearán para solucionar el problema planteado. Por último se describirá la configuración de estos algoritmos implementados en Matlab.

4.1. Problemas de optimización. Programación lineal

Un problema de optimización consta principalmente de tres componentes: un conjunto de incógnitas o variables a determinar, una función objetivo a ser minimizada o maximizada, y un conjunto de restricciones que deben ser satisfechas. Resolver este problema consiste en encontrar valores de las variables que optimizan (minimizar o maximizar) la función objetivo al tiempo que satisfacen todas las restricciones.

Existen problemas de optimización sin restricciones pero este trabajo se centra en un problema lineal con restricciones.

Los problemas de programación lineal son la clase más importante de problemas de optimización, base de resolución de otros problemas de optimización. Se definen como problemas de optimización lineales los problemas cuya función objetivo y

restricciones son funciones lineales (función polinómica de primer grado); es decir, una función cuya representación en el plano cartesiano es una línea recta.

$$\begin{aligned} &\text{minimizar:} && f(x) = c \cdot x \\ &\text{sujeto a:} && A \cdot x = 0 \quad x = (x_1, \dots, x_n) \\ & && b \cdot x \leq 0 \end{aligned}$$

La función objetivo es combinación lineal de las variables de decisión con una contaste que proporciona el coste asociado a esa variable. Las variables de decisión son variables continuas. Las restricciones con funciones lineales, combinación de las variables de diseño que cumplen con una condición de igualdad o de desigualdad.

Existe un caso de programación lineal en el que las variables son enteros binarios. Esto significa que las variables *x* solo pueden tener valores 0 o 1. Este tipo de problemas son los que permiten definir modelos de transporte, haciendo que las variables sean las rutas que se realizan (variable con valor 1) o no se realizan (variable con valor 0).

4.1.1. Conceptos básicos

Para caracterizar la solución obtenida de la optimización, se definen conceptos básicos como puntos vecinos, solución factible, mínimo local y mínimo global.

Definición 1.1 El conjunto de puntos vecinos de x, $N(x)$, es el conjunto de puntos ($x' \in X$) próximos al punto x en la variable espacial X, tal que $x' \in N(x)$ y $x \in N(x')$

Definición 1.2 Un punto $x \in X$ se considera un punto factible o solución factible si satisface todas las restricciones. Esto es, $h(x) = 0$ y $g(x) \leq 0$.

Definición 1.3 Un punto $x^* \in X$ se considera un mínimo local si x^* es un punto factible y si para cada punto factible $x \in N(x^*)$, $f(x^*) \leq f(x)$.

Evaluando las funciones objetivos de los puntos vecinos a x^*, siendo estos soluciones factibles, la función objetivo $f(x^*)$ tendrá un valor menor que el resto de evaluaciones de los puntos vecinos. Esto no significa que el resultado sea el óptimo del problema, significa que ha encontrado un mínimo local. Si el punto x^* no tiene vecinos, este será igualmente un mínimo local.

Definición 1.4 Un punto $x^* \in X$ se considera un mínimo global si x^* es un punto factible y si para cada punto factible $x \in X$, $f(x^*) \leq f(x)$.

La principal diferencia entre mínimo local y mínimo global (Figura 23) es que el local se compara con su conjunto de puntos factibles vecinos, y el mínimo global se compara con todo el conjunto de puntos.

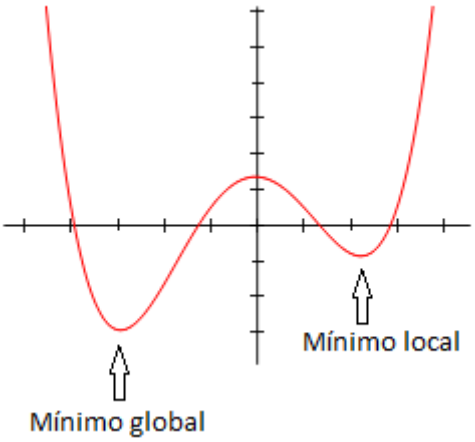


Figura 23.- Función con mínimo local y global

El resultado obtenido al solucionar este tipo de problemas puede ser uno de los cuatro posibles resultados que, por lo general, se obtienen en un problema de programación lineal. Éstos son:

1. *Solución única*. La función objetivo máxima interseca un solo punto.

2. *Soluciones alternativas*. El problema podría tener un número infinito de óptimos correspondientes a un segmento de línea.

3. *Solución no factible.* Es posible que el problema esté formulado de tal manera que no haya una solución factible. Esto puede deberse a que se trata de un problema sin solución o a errores en la formulación del problema. Lo último ocurre si el problema está sobrerrestringido, y ninguna solución satisface todas las restricciones.

4. *Problemas no acotados*. El problema está subrestringido y, por lo tanto, tiene límites abiertos. Como en el caso de la solución no factible, esto a menudo ocurre debido a errores cometidos durante la especificación del problema.

4.2. Notación y formulación del problema

Para formular de forma matemática este tipo de problemas es necesario definir una notación a modo de lenguaje común para transcribir el comportamiento del sistema explicado en los capítulos anteriores.

El problema consiste en un conjunto $\mathrm{R} = \{1,2, \dots, \mathrm{n}\}$ de n peticiones de viajeros y un conjunto $V = \{2n + 1,2n + ,2, \dots ,2n + v\}$ de *v* ofertas de servicio por parte de los

conductores, tal que $R \cap V = \emptyset$. Para cada petición del viajero $i \in R$ existe un punto de origen i y uno de destino $i + n$. Del mismo modo, para cada vehículo $k \epsilon V$ existe un punto de origen k y un punto de destino $k + v$, y una variable que define la distancia máxima de desvío que está dispuesto a realizar cada conductor MTD_k. Se usará $P = \{1, \ldots, n\}$ para agrupar todos los puntos de recogida de los viajeros y $D = \{n + 1, \ldots, 2n\}$ para agrupar todos los puntos de entrega de los viajeros. El conjunto de todos los puntos de recogida y entrega se denota como $N = P \cup D$ y el conjunto de todos los puntos del sistema incluyendo los orígenes y destinos de los vehículos se denota como $A = N \cup \{k, k + v\} \quad \forall k \in V$.

En cuanto a la función objetivo, existe una variable $d_{i,j}$ que denota la distancia de viaje directa entre los puntos i y j. Las variables de decisión $x_{i,j}^k$ serán variables binarias con valor 1 si el vehículo k entra en el punto i y viaja directamente al punto j, y con valor 0 si no realiza esta ruta. La función objetivo debe tener en cuenta la maximización de usuarios que se emparejan y la minimización de la distancia total que se recorre. Para ello se definen las variables de ponderación α y β que definen la importancia relativa de los distintos componentes de la función objetivo. También es necesario crear la variable r que hace referencia al número de viajeros que se han emparejado, y n que es la constante que determina el número máximo de emparejamientos (en este problema será el número de viajeros el sistema)

La siguiente tabla resume la notación descrita anteriormente, para facilitar el estudio de las funciones que se describirán a continuación:

$\boldsymbol{R}$	Número de puntos de viajero
$\boldsymbol{V}$	Número de puntos de conductor
$\boldsymbol{i}$	Punto de recogida de un viajero
$\boldsymbol{i + 1}$	Punto de entrega de un viajero
$\boldsymbol{k}$	Punto de origen de un conductor
$\boldsymbol{k + 1}$	Punto de destino de un conductor
$\boldsymbol{P}$	Conjunto de puntos de recogida de los viajeros
$\boldsymbol{D}$	Conjunto de puntos de entrega de los viajeros
$\boldsymbol{N}$	Conjunto de puntos de de viajero
$\boldsymbol{A}$	Conjunto del total de puntos del problema
$\boldsymbol{x_{i,j}^k}$	Ruta en el vehículo *k* entre los puntos *i* y *j*
$\boldsymbol{d_{i,j}}$	Distancia entre los puntos *i* y *j*
A	Coeficiente de ponderación de distancia
B	Coeficiente de ponderación de número de emparejamientos
$\boldsymbol{n}$	Número máximo de emparejamientos posible
$\boldsymbol{r}$	Número de emparejamientos realizados
α	Coeficiente de ponderación de la distancia
β	Coeficiente de ponderación del número de emparejamientos

Se puede entonces plantear la función objetivo de este sistema como:

$$\min \propto \sum_{k \in V} \sum_{i,j \in A} x_{i,j}^{k} \cdot d_{i,j} + \beta \cdot (n - r)$$

Esta función multiplica todas las rutas que se den en el sistema por la distancia asociada a esta. Este valor se multiplica a su vez por $\propto$ que pondera la importancia de la distancia total sobre el resultado final del problema. El segundo término es la resta de las rutas máximas que se pueden realizar menos las que se realizan. También va multiplicado por **β**, el factor de ponderación que marca la importancia de este término sobre el resultado final del problema.

Del mismo modo, a continuación se formulan las distintas restricciones del sistema.

Restricción 1: Cada vehículo que abandona su origen visita un único punto de recogida de un usuario o viaja directamente a su destino.

$$\sum_{j \in P \cup \{k+v\}} x_{k,j}^{k} = 1, \qquad \forall k \epsilon V$$

Restricción 2: Cada vehículo que llega a su destino, parte de un único punto de entrega de un usuario o viaja directamente desde su destino.

$$\sum_{j \in D \cup \{k\}} x_{i,k+v}^{k} = 1, \qquad \forall k \epsilon V$$

Restricción 3: Cada usuario se empareja una única vez, con un único vehículo.

$$\sum_{k \in V} \sum_{j \in A} x_{i,j}^{k} \leq 1, \qquad \forall i \epsilon P$$

Restricción 4.a: El vehículo que recoge al viajero, debe entregarlo en su destino.

$$x_{k,i}^{k} - x_{i,i+n}^{k} = 0, \qquad \forall k \epsilon V, \forall i \epsilon P$$

Restricción 4.b: El vehículo que entrega al viajero, debe entregarlo en su destino.

$$x_{i,i+n}^{k} - x_{i+n,k+v}^{k} = 0, \qquad \forall k \epsilon V, \forall i \epsilon D$$

Restricción 5: El conductor está dispuesto a realizar una ruta que no supere un valor máximo.

$$\sum_{i,j\in A} x_{i,j}^{k} \cdot d_{i,j} \leq MTD_k\,, \qquad \forall k \epsilon V$$

Por lo que el modelo completo tiene se define como:

Función objetivo:

$$\min \propto \sum_{k\in V} \sum_{i,j\in A} x_{i,j}^{k} \cdot d_{i,j} + \beta \cdot (n - r)$$

Sujeto a:

$$\sum_{j\in P\cup\{k+v\}} x_{k,j}^{k} = 1, \qquad \forall k \epsilon V$$

$$\sum_{j\in D\cup\{k\}} x_{i,k+v}^{k} = 1, \qquad \forall k \epsilon V$$

$$\sum_{k\in V} \sum_{j\in A} x_{i,j}^{k} \leq 1, \qquad \forall i \epsilon P$$

$$x_{k,i}^{k} - x_{i,i+n}^{k} = 0, \qquad \forall k \epsilon V, \forall i \epsilon P$$

$$x_{i,i+n}^{k} - x_{i+n,k+v}^{k} = 0, \qquad \forall k \epsilon V, \forall i \epsilon D$$

$$\sum_{i,j\in A} x_{i,j}^{k} \cdot d_{i,j} \leq MTD_k\,, \qquad \forall k \epsilon V$$

$$x_{i,j}^{k} \in \{0,1\}, \qquad \forall i,j \epsilon A$$

4.2.1. Ventanas temporales

Para añadir las ventanas temporales es necesario ampliar el número de restricciones del modelo descrito en el apartado 4.2 de esta memoria con las siguientes funciones en las que se tiene en cuenta las restricciones temporales:

Restricción 6: Si se realiza una ruta, la hora de llegada tiene que ser mayor que la de salida más el tiempo de viaje para evitar incoherencias temporales. De esta manera se garantizaría la continuidad temporal de la ruta.

$$x_{i,j}^{k} \cdot \left(T_i^k - t_{i,j} - T_j^k\right) \leq 0, \qquad \forall k \epsilon V, i, j \epsilon A$$

Restricción 7: El valor de la hora de salida/llegada del nodo debe estar acotado por un tiempo máximo y mínimo.

$$a_i \leq T_i^k \leq b_i \qquad \forall k \epsilon V, \forall i \epsilon A$$

Donde:

$\boldsymbol{T_i^k}$	Hora de salida de un punto de partida
$\boldsymbol{T_j^k}$	Hora de llegada a un punto de destino
$t_{i,j}$	Tiempo de viaje entre dos puntos
$\boldsymbol{a_i}$	Hora mínima de salida/llegada del nodo i
$\boldsymbol{b_i}$	Hora máxima de salida/llegada del nodo i
$x_{i,j}^k$	Ruta realizada en el vehículo k entre los puntos i y j

La restricción 6 establece una relación entre dos variables, una de tipo entero y otra de tipo entero binario. Esto hace que el problema se convierta en un problema de enteros mixtos (enteros y binarios) y pierda la linealidad (problema no lineal). Este tipo de problemas denominados MINLP (del inglés, Mixed Integer Nonlinear Programming) es un problema de compleja solución ya que combina la dificultad de trabajar con dos tipos de enteros y la no linealidad, como se ha descrito en el apartado 0.

4.3. Algoritmo *branch and bound*

El algoritmo busca una solución óptima para el problema de programación binaria entera resolviendo una serie de problemas LP-aproximados, en los cuales el requisito de las variables binarias enteras se sustituye por una restricción menos exigente en la que 0 ≤ x ≤ 1. El algoritmo:

- Busca una solución factible entera binaria.
- Actualiza el mejor punto binario entero encontrado hasta ese punto mientras el árbol crece.
- Verifica que hay una solución factible entera solucionando una serie de problemas de programación lineal.

A continuación se describe con más detalle el algoritmo del método branch-and-bound.

El algoritmo crea un árbol de búsqueda añadiendo restricciones al problema, a esto se le llama ramificación o "branching". En el paso de ramificación, el algoritmo escoge la variable x_j cuyo valor no está definido y añade la restricción $x_j = 0$ para crear una rama del árbol y $x_j = 1$ para formar otra. Este proceso se puede representar mediante un arbol binario en el que los nodos representan estas restricciones añadidas. La a continuación muestra un árbol completo para un problema con tres variables, $x_1, x_2\ y\ x_3$.

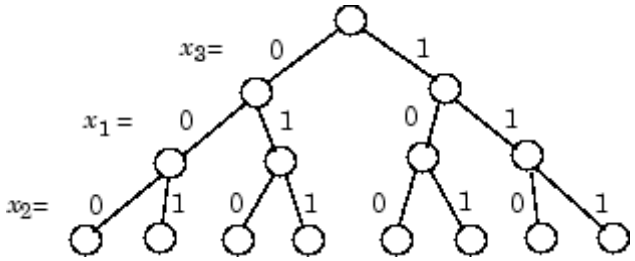


Figura 24.- Árbol de optimización

La idea clave del algoritmo de ramificación y poda es: si la menor rama para algún árbol nodo (conjunto de candidatos) A es mayor que la rama padre para otro nodo B, entonces A debe ser descartada con seguridad de la búsqueda. Este paso es llamado poda, y normalmente es implementado manteniendo una variable global *m* que guarda el mínimo nodo padre visto entre todas las subregiones examinadas hasta entonces. Cualquier nodo cuyo nodo hijo es mayor que *m* puede ser descartado. La recursión para cuando el conjunto candidato S es reducido a un solo elemento, o también cuando el nodo padre para el conjunto S coincide con el nodo hijo. De cualquier forma, cualquier elemento de S va a ser el mínimo de una función dentro de S.

En cada nodo, el algoritmo soluciona el problema de tipo LP usando las restricciones en cada nodo y decide ramificarse o avanzar por esa rama hacia otro nodo, dependiendo del resultado. Hay tres posibilidades.

- Si el problema LP-aproximado en el nodo actual no es factible o su valor óptimo es mayor que el mejor punto entero calculado, el algoritmo borra ese nodo del árbol, después de lo cual deja de buscar en ramas por debajo de ese nodo.

- Si el algoritmo encuentra una solución factible con un valor de la función objetivo menor que el mejor punto hasta el momento, actualiza el valor del mejor punto y se mueve al siguiente nodo.

- Si el problema LP-aproximado es óptimo pero no entero, y el valor de la función objetivo del problema LP-aproximado es menor que el mejor punto hasta el momento, el algoritmo se ramifica.

La solución del problema LP-aproximado crea un límite inferior para el problema de programación entera binaria. Si la solución del problema LP-aproximado es un vector binario, se proporciona un límite superior para el problema de programación binaria.

Conforme el árbol genera más nodos, el algoritmo actualiza los límites superior e inferior en la función objetivo, usando los límites obtenidos en la etapa de limitación. El límite en valor objetivo sirve como umbral para eliminar ramas innecesarias.

4.4. Algoritmo genético

Los algoritmos genéticos son un método para solucionar problemas de optimización con y sin restricciones que está basado en la selección natural, el proceso que rige la evolución biológica. El algoritmo modifica repetidamente un conjunto de soluciones individuales, llamado población. En cada iteración, el algoritmo selecciona individuos de forma aleatoria de la población actual para ser parientes y los usan para generar los hijos de la siguiente generación, la siguiente población. Tras sucesivas generaciones, la población “evoluciona” hacia la solución óptima. Se puede aplicar el algoritmo genético para problemas que no se ajustan a los algoritmos de optimización convencionales, incluyendo problemas en los que la función objetiva es discontinua, no diferenciable, estocástica o altamente no lineal. El algoritmo genético puede servir para problemas de enteros mixtos, para los que algunos componentes están restringidos a valores enteros.

El algoritmo aplica tres tipos de reglas en cada iteración para crear la nueva generación a partir de la población actual:

- ***Reglas de selección***: Selecciona los individuos, denominados parientes que pasarán a formar parte de la población en la siguiente generación.

- ***Reglas de cruce***: Combina dos parientes para formar un hijo para la siguiente generación.

- ***Reglas de mutación***: Aplica cambios aleatorios a parientes para crear hijos.

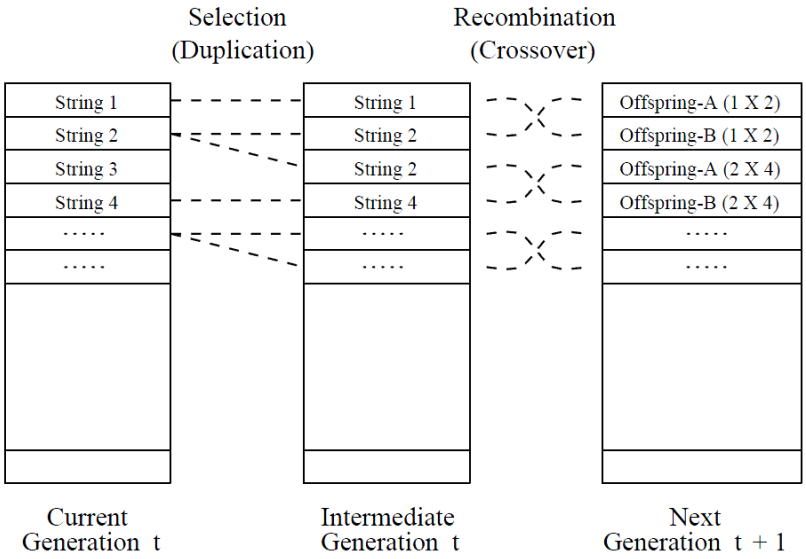


Figura 25.- Selección y cruce del algoritmo genético

En la Figura 25 se muestra un ejemplo de evolución de la población. En el primer paso se selecciona los mejores individuos de la población, estos son los que menor valor han devuelto al evaluar la función *fitness.* Tras este proceso se pasa a cruzar a los mejores individuos para generar la siguiente generación.

El algoritmo genético se diferencia de los algoritmos clásicos de optimización, basados en derivaciones, en dos conceptos que se describen en la Tabla 5.

Tabla 5.- Diferencias entre algortimos clásicos de optimización y algoritmos genéticos

Algoritmo clásico	Algoritmo genético
Genera un único punto en cada iteración. La secuencia de puntos se aproxima a la solución óptima.	Genera una población de puntos en cada iteración. El mejor punto de la población se aproxima a la solución óptima.

Selecciona el siguiente punto de la secuencia mediante métodos deterministicos de computación.	Selecciona la siguiente población mediante un procesado en el que se emplean generadores de números aleatorios.

4.5. Toolbox de optimización de Matlab

Matlab es un lenguaje técnico computacional y un ambiente interactivo para el desarrollo de algoritmos, visualización y análisis de datos, y computación numérica. Es usado en una gran variedad de aplicaciones, como el procesamiento de señales e imágenes, comunicaciones, modelado y análisis financiero, entre otros. Se pueden añadir toolboxes, los cuales son colecciones de funciones para un propósito en especial, que extienden las funcionalidades de Matlab. Uno de estos, es el toolbox de Optimización, el cual provee algoritmos para resolver diferentes tipos de problemas de optimización, con restricciones o sin ellas; los cuales incluyen programación lineal, programación cuadrática, minimización (maximización) de funciones de una o varias variables, solución de sistemas de ecuaciones no lineales, entre otros.

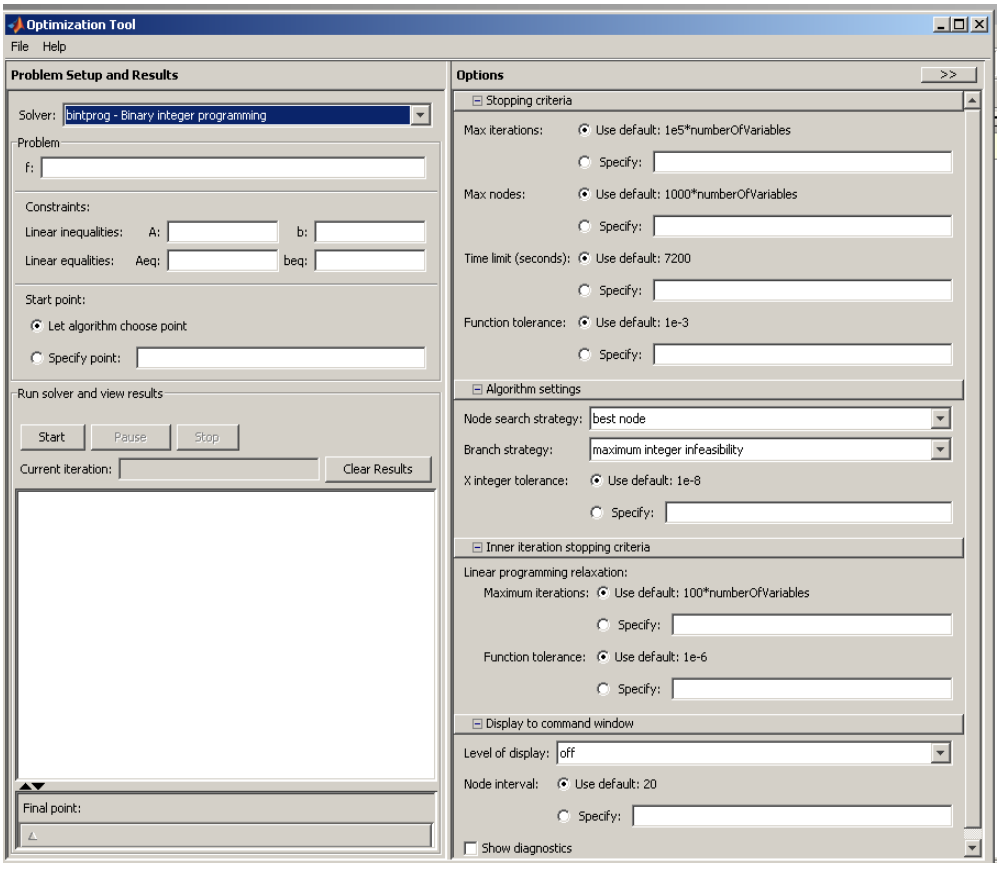


Figura 26.- Optimization toolbox de Matlab

El *toolbox* de optimización de Matlab (Figura 26) es un conjunto de funciones (almacenadas en archivos m-files, estándar de programación propio de Matlab) que proveen algoritmos para solucionar diferentes tipos de problemas. Además, posee una interfaz gráfica que facilita a los usuarios menos experimentados usar fácilmente el *toolbox*, sin necesidad de hacer uso de código.

4.5.1. Algoritmo bintprog

bintprog usa un algoritmo para programación lineal basado en el algoritmo de "branch-and-bound" o "ramificación y poda" para solucionar problemas de programación binaria. La sintaxis para hacer uso de la función *bintprog* es:

x = bintprog(f,A,b,Aeq,beq,x0,options)

Donde:

- f: es el vector de coeficientes de la función objetivo, organizado según las variables.

- A, b: corresponden a las restricciones de desigualdad, siendo el primero la matriz y el segundo el vector del lado derecho del sistema de inecuaciones $A \cdot x \leq b$.
- Aeq, beq: tienen el mismo tratamiento que A y b, respectivamente, teniendo en cuenta que los nuevos corresponden a un sistema de ecuaciones, en tanto que los antiguos constituyen uno de inecuaciones.

- x0: es el punto inicial para la iteración. Según el algoritmo usado es posible, o no, omitir este último.

La programación entera binaria es el problema de encontrar un vector binario x que minimiza la función lineal f^T sujeta a restricciones lineales:

$$\min_x f^T x$$

$$\text{Sujeto a: } A \cdot x \leq b, A_{eq} \cdot x = b_{eq}, x \in \{0,1\}$$

4.5.2. Algoritmo genético

El algoritmo genético de Matlab debe ser configurado para su correcto funcionamiento, por lo que es necesario conocer los parámetros de éste y la influencia de estos en la solución obtenida.

La función de adecuación (o *fitness function de su término en inglés) es la función que se quiere optimizar*. Para los problemas estándar de optimización, esta función se conoce como función objetivo.

Un individuo es cualquier punto al que se le puede aplicar la función *fitness*. El valor de esta función es la puntuación del individuo evaluado. Por ejemplo, si la función *fitness* es:

$$f(x_1, x_2, x_3) = (2x_1 + 1)^2 + (3x_2 + 4)^2 + (x_3 - 2)^2,$$

el vector (2, -3, 1), con un número de datos igual que el número de variables del problema, es un individuo. La puntuación de este individuo será *f*(2, –3, 1) = 51. Un individuo pude denominarse genoma y al vector de enteros del individuo, genes.

Una población es una matriz de individuos. Por ejemplo, si el tamaño de población es de 100 y el número de variables de la función objetivo es 3, la población se representa con una matriz de 100 por 3. El mismo individuo puede aparecer más de una vez en cada población.

En cada iteración, el algoritmo genético procesa la población actual creando una nueva generación. Cada generación sucesiva se denomina nueva generación.

La diversidad hace referencia a la distancia media entre individuos de una población. Si una población es altamente diversa, la distancia media es grande, del contrario es pequeña. La siguiente figura muestra una población en la parte izquierda del gráfico con alta diversidad (de color azul) y otra población en la parte derecha con baja diversidad (de color rojo).

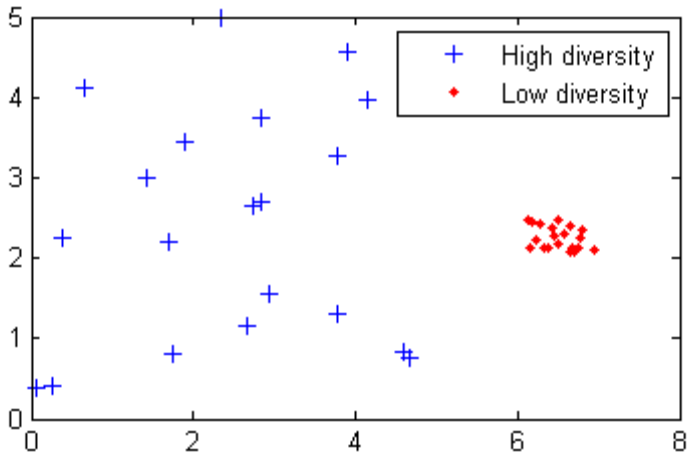


Figura 27.- Diversidad de poblaciones

La diversidad es esencial para el algoritmo genético porque permite que este busque en una región más amplia del espacio.

El valor *fitness* de un individuo es el valor obtenido al evaluar la función *fitness* para ese individuo. Dado que la herramienta busca el mínimo valor de la función *fitness* , el mejor valor de *fitness* para una población será el menor para cada individuo de la población.

Para crear una nueva generación, el algoritmo genético selecciona ciertos individuos de la población actual, llamados parientes, y los usa para crear nuevos individuos en la siguiente generación, llamados hijos. El algoritmo selecciona parientes que tienen mejor función *fitness.*

A continuación se resume el funcionamiento del algoritmo genético:

1. El algoritmo comienza creando una población inicial aleatoria, o se puede suministrar una población inicial.

2. Después, el algoritmo crea una secuencia de nuevas poblaciones. En cada iteración. El algoritmo emplea individuos de la generación actual para crear la siguiente generación. Para crear la nueva población, el algoritmo realiza los siguientes pasos:

    a. Evalúa cada miembro de la población actual calculando su función *fitness.*

    b. Ordena los valores obtenidos de la evaluación para tener un rango ordenado de valores, para un mejor manejo de los datos.

c. Selecciona individuos, llamados parientes, basándose en su función *fitness*.

d. Algunos de los individuos de la población actual con el menor valor de *fitness* son elegidos como élite. Esta élite pasa directamente a la siguiente población.

e. Genera hijos de los parientes. Los hijos se generan provocando () cambios aleatorios en uno de los parientes (*mutación*) o bien combinando los vectores de un par de parientes (*cruce o crossover*).

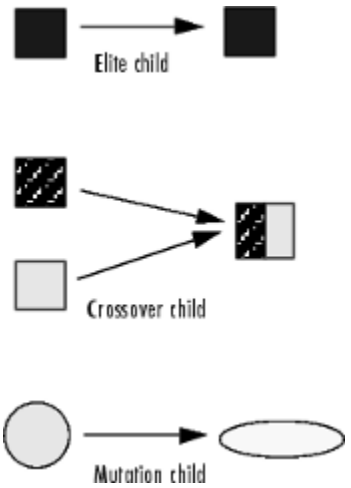


Figura 28.- Estrategias de evolución del algoritmo genetico

f. Sustituye la población actual con los hijos creados para formar la siguiente generación.

3. El algoritmo para cuando se alcanza algún criterio de parada como puede ser un número máximo de generaciones, una restricción de tipo temporal, un estancamiento de variación en la función *fitness*.

Ejemplo: Problema del viajero aplicado a Estados Unidos

El ejemplo propuesto muestra el problema del viajero. En este caso el viajero debe visitar cada uno de los 50 estados una única vez empleando la ruta más corta.

Población inicial

El algoritmo crea inicialmente una población aleatoria, como se muestra en la siguiente figura. El algoritmo está programado para mostrar el mejor individuo. En la se puede observar que la primera población no cumple con las restricciones. Esto se debe a que la población inicial es aleatoria. Si se conoce una posible solución, se puede alimentar el algoritmo con estos datos considerándolos la población inicial.

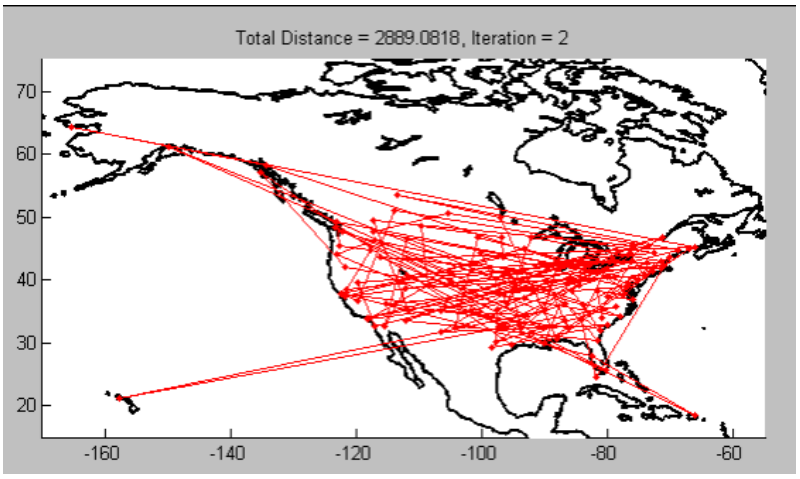


Figura 29.- Población inicial del problema

Tras evolucionar el algoritmo en sucesivas iteraciones la solución va mejorando hasta dar con una que cumple con los criterios de parada.

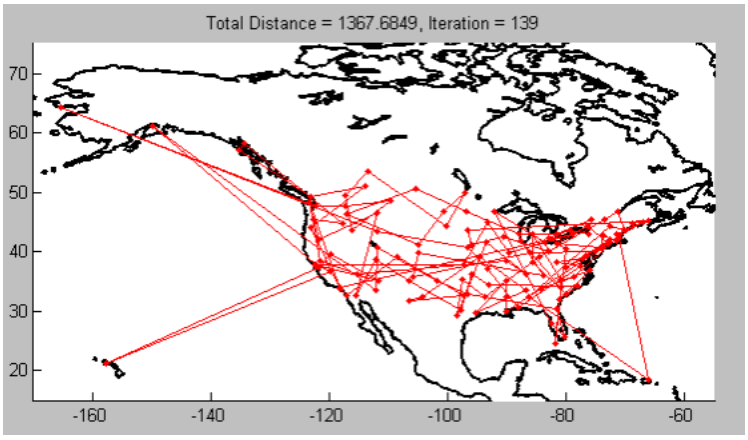


Figura 30.- Problema TSP - Iteración 139

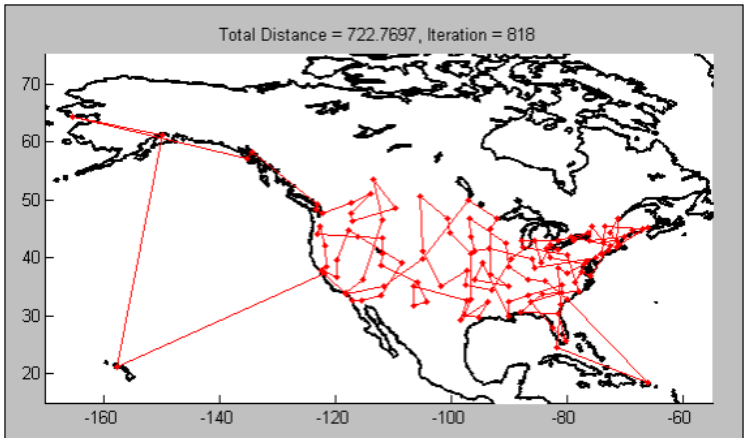


Figura 31.- Problema TSP - Iteración 818

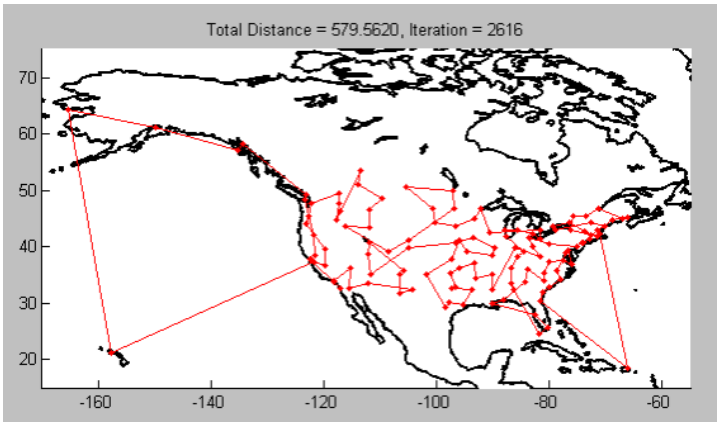


Figura 32.- Problema TSP - Iteración 2616

El algoritmo tiende a minimizar la función objetivo, pero este tipo de métodos no garantiza que el resultado sea siempre el óptimo como lo hacen los métodos deterministas. La solución de este algoritmo es buena, pero no siempre la mínima.

4.5.3. Métodos MINLP

Matlab no tiene una función de optimización para este tipo de problemas pero permite resolver problemas MINLP gracias a un Toolbox externo al propio programa llamado OPTI TOOLBOX. Este provee de una serie de solvers para distintos tipos de problemas de optimización (MILP, QP, MIQP, …) entre los que se encuentra el solver de MINLP, NOMAD [10].

NOMAD usa un tipo de algoritmo de búsqueda llamado Mesh Adaptive Direct Search para resolver problemas no lineales y no diferenciables. El software suministrado, es de distribución libre y permite solucionar problemas de unos pocos centenares de variables.

4.6. Descripción del software desarrollado

La estructura del programa que se ha desarrollado para revisar el problema se muestra de forma resumida en el siguiente diagrama de flujo.

En la parte de generación del entorno, se crean una matriz de distancias (definida al final de esta página) basada en una serie de puntos generados aleatoriamente. Una matriz de distancias es una matriz cuyos elementos representan las distancias entre los puntos, tomados por pares, de un conjunto. Se trata, por lo tanto, de una matriz simétrica de tamaño *NxN* (dado un conjunto de *N* puntos en el espacio euclídeo) conteniendo números reales no negativos como elementos. El número *N* de pares de puntos, *(N-1)/2*, es el número de elementos independientes en la matriz de distancias.El número de puntos que se crea es de $2 \cdot (numero\ de\ conductores + número\ de\ viajeros)$.

$$\begin{bmatrix} 0 & d_{1,2} & \dots & d_{1,j} \\ d_{2,1} & 0 & \dots & d_{2,j} \\ \dots & \dots & \ddots & \dots \\ d_{i,1} & d_{i,2} & \dots & 0 \end{bmatrix}$$

La matriz de distancias tiene la particularidad de que es simétrica respecto su diagonal principal (formada por ceros), lo que implica que $d_{i,j} = d_{j,i}$.

Tras generar la matriz de distancias con los puntos que van a formar parte del entorno se pasa a generar los datos de usuario (Figura 33). Estos se generan como se muestra a continuación.

	1	2	3	4	5
1	'ID'	'R/D'	'INI'	'FIN'	'MTD'
2	1	1	1	15	380.7887
3	2	1	2	16	509.9020
4	3	1	3	17	100
5	4	1	4	18	400
6	5	0	5	19	100
7	6	0	6	20	70.7107
8	7	0	7	21	509.9020
9	8	0	8	22	291.5476
10	9	0	9	23	50
11	10	0	10	24	813.9410
12	11	0	11	25	854.4004
13	12	0	12	26	180.2776
14	13	0	13	27	250
15	14	0	14	28	701.7834

Figura 33.- Usuarios generados

Cada participante tiene un identificador (primera columna), un rol definido (segunda columna, si el valor es 1 son conductores y si es 0 son viajeros), un punto de origen (tercera columna) que es el índice de uno de los puntos que se han creado anteriormente, un punto de destino (cuarta columna) que es otro índice como el descrito anteriormente y una distancia máxima de viaje (quinta columna).

Una vez generado el mapa y los usuarios se mostrará en el mapa total con los desplazamientos de los distintos participantes (de rojo los conductores y de verde los viajeros, Figura 34).

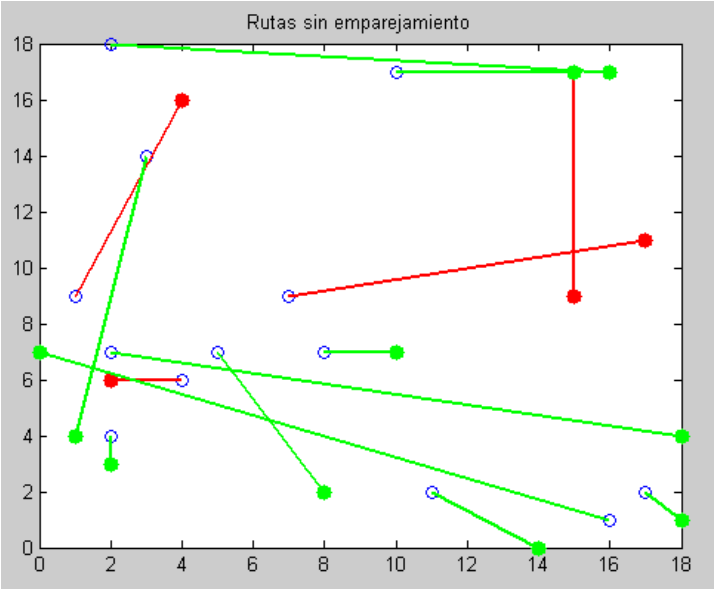


Figura 34.- Escenario generado aleatoriamente

El siguiente paso será crear las restricciones, para lo que será necesario procesar los datos de manera que se reproduzcan en vectores las restricciones descritas anteriormente. Para ello se han propuesto dos métodos distintos de procesamiento de los datos, que se describirán en las secciones 4.7.1 y 4.7.2 . Tras generar las restricciones se lanza el algoritmo de optimización seleccionado.

A continuación, en la Tabla 6, se recoge una descripción de cada uno de las funciones y estructuras de datos desarrollados y generados, respectivamente.

Tabla 6.- Descripción del software desarrollado

Fichero	Descripción
dist.mat	Contiene la matriz de distancia
puntos.mat	Contiene los puntos
users.mat	Contiene los datos de cada participante
distan.m	Calcula la distancia entre dos puntos,
gentabdist	Genera, de forma aleatoria, una matriz de distancias de NxN elementos.
numervar.m	Calcula el número de variables que se generan en el sistema.
rideshare.m	Función principal. Llama las distintas funciones descritas en esta tabla para generar los datos del problema y lanzar los algoritmos de optimización.
usuarios.m	Genera los datos de los usuarios (rol, puntos de inicio y fin)

4.7. Limitaciones

Debido al alto número de enteros binarios que se van a generar, es necesario considerar el tamaño de las matrices ya que puede entrar en conflicto con la memoria que Matlab asigna al valor máximo de matriz.

El máximo tamaño de matriz posible viene determinada por el tamaño del mayor espacio contiguo de bloque de memoria libre. Esto implica que será una limitación para el número de elementos que Matlab podrá manejar en el problema. MATLAB calcula este número del mínimo de estos dos valores:

- El mayor bloque de memoria contigua encontrado por el espacio virtual de direcciones de Matlab.

- El total de memoria disponible del sistema.

Este valor se puede consultar mediante el comando *memory* de Matlab. Para este proyecto se ha usado un LG E500 (2.41 GHz, 3 GB de RAM) con Windows XP SP 3.

```
>> user = memory
user =
   MaxPossibleArrayBytes: 73531392
   MemAvailableAllArrays: 1.5241e+09
        MemUsedMATLAB: 386527232
```

Todos los valores son de tipo double-precision (64 bits) y las unidades son en bytes. El mayor número de elementos que puede contener una matriz de *doubles* es de aproximadamente 90.000.000 de elementos. Esto puede entrar en conflicto debido al alto número de variables que se generan en este tipo de problemas conforme aumenta el número de participantes

4.7.1. Primera aproximación

La primera estructura de datos propuesta se basa en la formulación de los distintos métodos en los que el vector resultado considera todas las posibles rutas entre todos los puntos del sistema. Para ello el vector con las variables de diseño (vector con la solución) contendrá el mismo número de elementos que la matriz de distancia. Esto incluye las mismas rutas en sentidos contrarios y rutas que no deberían realizarse.

Se opta por este primer método para validar el modelo viendo su correcto funcionamiento para entornos simulados. Para ello se evalúan una serie de condiciones

que se deben cumplir. Para este problema se han considerado las siguientes condiciones:

- **Evitar rutas repetidas:** De esta manera se evita que se dé una ruta x, y su simétrica en la matriz de distancias.

- **Evitar rutas inválidas:** Este tipo de rutas son, por ejemplo, rutas entre el punto de origen de un conductor hacia otro punto de origen de otro conductor distinto.

- **Evitar rutas de distancia 0:** Esta ruta se daría si el conductor va de su punto de origen a su punto de origen, ruta con una distancia asociada de 0.

Para validar el modelo se busca que el algoritmo de optimización devuelva una solución factible. En este primer paso no se tiene en cuenta el número de rutas que se emparejan, sino que el resultado del emparejamiento sea el correcto. Se pasa una serie de pruebas con distintos número de conductores y de viajeros y se observan los resultados.

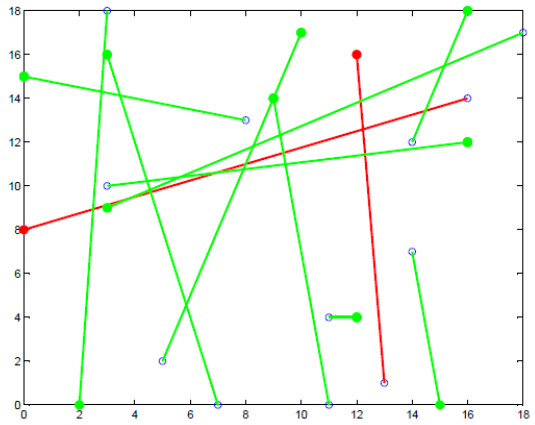


Figura 35.- Escenario con dos conductores y diez viajeros

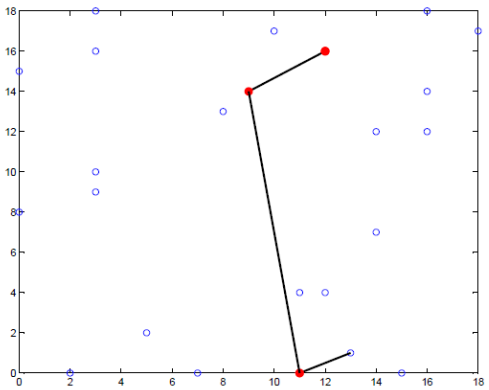


Figura 36.- Emparejamiento del primer conductor

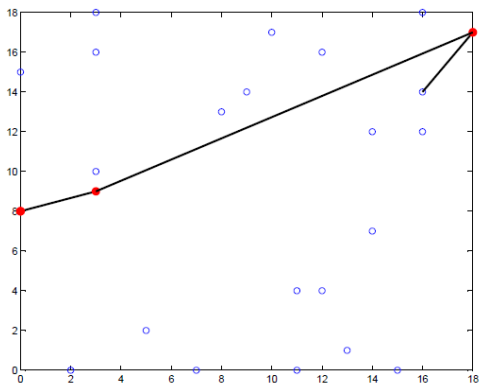


Figura 37.- Emparejamiento del segundo conductor

Se crea un entorno en el que la relación vehículos/pasajeros es baja (en este ejemplo 1:5). El entorno cuenta con 2 vehículos, de color rojo en la Figura 35, y 10 viajeros en sus cercanías. El coeficiente de emparejamiento β es 1000 veces mayor que el de distancia α. Esto provocará que el algoritmo fuerce los emparejamientos, ya que da mayor importancia al emparejamiento que a la distancia recorrida (este problema será estudiado más adelante). Estas pruebas demuestran que las rutas se suceden de forma adecuada (Figura 36 y Figura 37), partiendo de un origen de un conductor para recoger a un viajero, llevarle a su destino e ir al destino final del conductor.

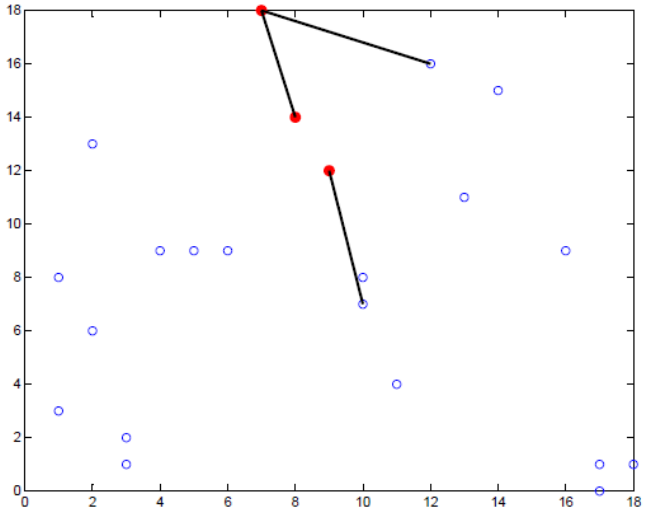


Figura 38.- Ruta inválida

La Figura 38 muestra un caso de ruta inválida que se produjo durante la depuración del modelo, en la que el conductor enlaza la ruta en dos tramos (parte superior) pero aparece un tercer tramo que compromete la continuidad de la ruta. Este tipo de errores se ha eliminado, y el modelo funciona como es debido.

Conforme a la formulación del problema, será necesario crear variables de diseño de número de elementos igual al número de elementos de la matriz de distancias. Esto provoca que el número de variables se dispare de manera exponencial.

Se puede determinar el número de variables que habrá en las restricciones mediante la función:

$$f(x,y) = (3 \cdot x + y + 2 \cdot x \cdot y) \cdot 2 \cdot x^2 \cdot y$$

Donde:

$$x: Número\ de\ conductores$$
$$y: Número\ de\ viajeros$$

El término entre paréntesis establece el número total de restricciones que tendrá el problema y el término fuera del paréntesis establece el número de elementos de cada restricción (el número de elementos de la matriz de distancia multiplicado por el número de vehículos). Evaluando esta función en ambas variables se puede observar que la función crece exponencialmente, conforme aumenta el número de participantes (Figura 39 y Figura 40).

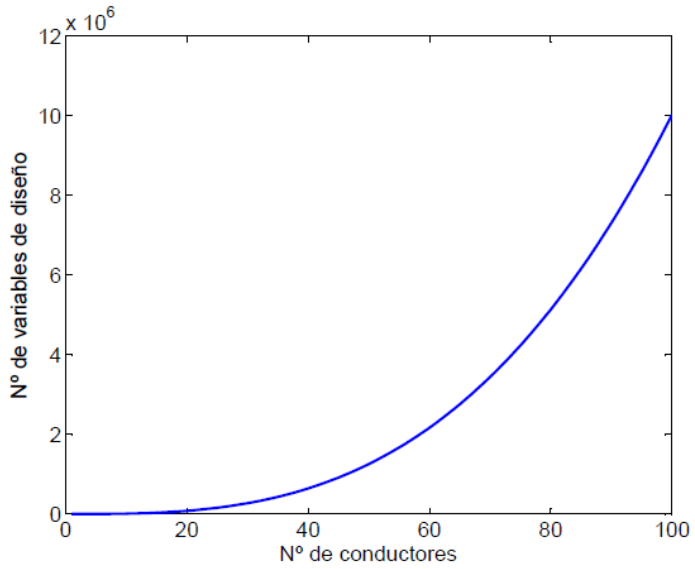


Figura 39.- Relación de variables de diseño y nº de conductores

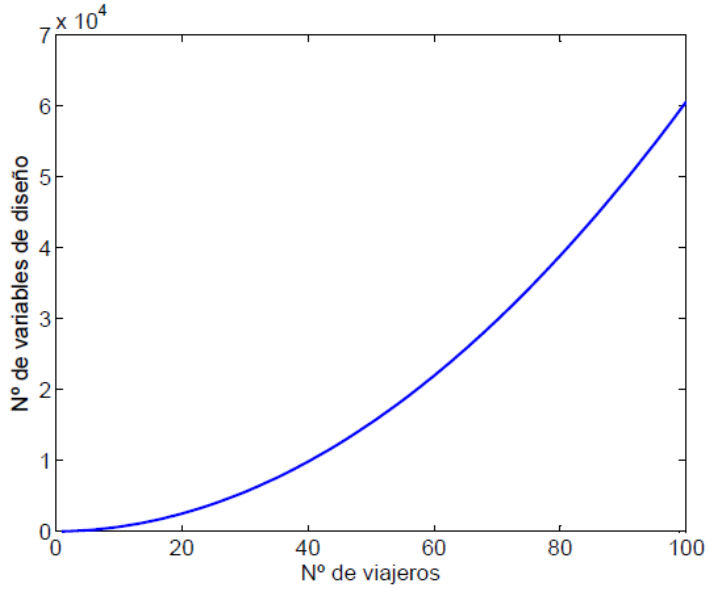


Figura 40.- Relación de variables de diseño y nº de viajeros

La Figura 41 muestra la representación espacial de la función evaluada entre x = [0,100] e y = [0,100]. Se observa que la dependencia del número de elementos es mayor con los conductores que con los viajeros. Para 100 conductores se generan del orden de 10^7 elementos, para un mismo número de viajeros (se ha usado únicamente 1 viajero ya que la intención de estas gráficas es la de mostrar la magnitud del problema).

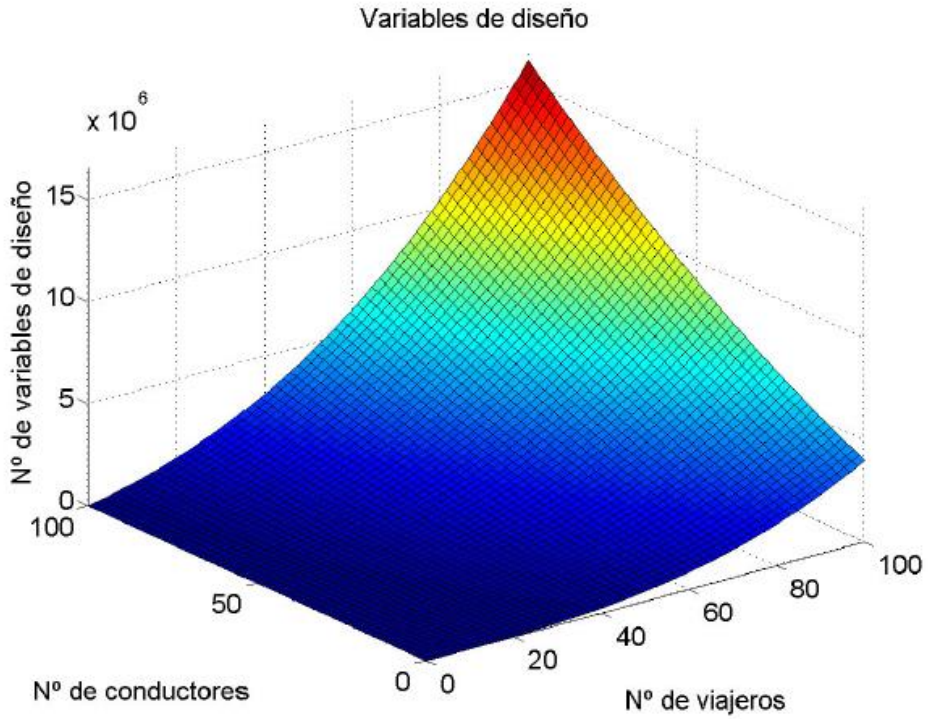


Figura 41.- Representación tridimensional de la relación de variables con el número de participantes

4.7.2. Segunda aproximación

Dado que el método anterior provoca un número alto de variables que no aportan valor al modelo, excepto en el momento de la validación, fue necesario reducir el tamaño de las matrices eliminando esa información redundante o no necesaria. Cabe destacar que se alcanzaban las limitaciones de memoria de Matlab para un número no demasiado grande de participantes, por lo que es necesario un modelo más "*ligero*".

La filosofía que se ha seguido para reducir el número de elementos del sistema se ha basado en reducir el número de elementos de las restricciones, ya que el número de restricciones sigue siendo el mismo. La notación del vector que se emplea para esta versión del modelo es la siguiente:

$$1d(i)3r(f)$$

Se identifica primero el conductor mediante su número identificativo y una letra *d* que lo acompaña. Después se identifica el viajero mediante su número identificativo y una letra *r* que lo acompaña. Entre paréntesis se indica si el punto es de inicio mediante una i y si es de final mediante con una f. El paréntesis se coloca a la derecha del participante al que corresponda.

La estructura de datos que se ha propuesto para disminuir el número de variables de diseño es la que se describe a continuación:

Ruta directa del origen al destino del conductor

$$\begin{cases} 1d(i,f) \\ 2d(i,f) \\ . \\ . \\ . \\ Kd(i,f) \end{cases}$$

Ruta del punto de recogida al punto de entrega de cada viajero en cada coche

$$\begin{cases} 1d1r(i,f) \\ 1d2r(i,f) \\ . \\ . \\ 2d1r(i,f) \\ 2d2r(i,f) \\ . \\ . \\ KdNr(i,f) \end{cases}$$

Rutas desde el origen de los conductores hacia el punto de recogida de los viajeros

$$\begin{cases} 1d(i)1r(i) \\ 1d(i)2r(i) \\ . \\ . \\ 2d(i)1r(i) \\ 2d(i)2r(i) \\ . \\ . \\ Kd(i)Nr(i) \end{cases}$$

Rutas desde el punto de entrega de los viajeros hacia el punto de destino de los conductores

$$\begin{cases} 1d(f)1r(f) \\ 1d(f)2r(f) \\ . \\ . \\ 2d(f)1r(f) \\ 2d(f)2r(f) \\ . \\ . \\ Kd(f)Nr(f) \end{cases}$$

Del mismo modo que para el caso anterior, se pasa a analizar el número de variables que se genera con esta simplificación. Se puede determinar el número de variables que habrá en las restricciones mediante la función:

$$f(x,y) = (3 \cdot x + y + 2 \cdot x \cdot y) \cdot (x + 3 \cdot x \cdot y)$$

Donde:

$$x: Número\ de\ conductores$$
$$y: Número\ de\ viajeros$$

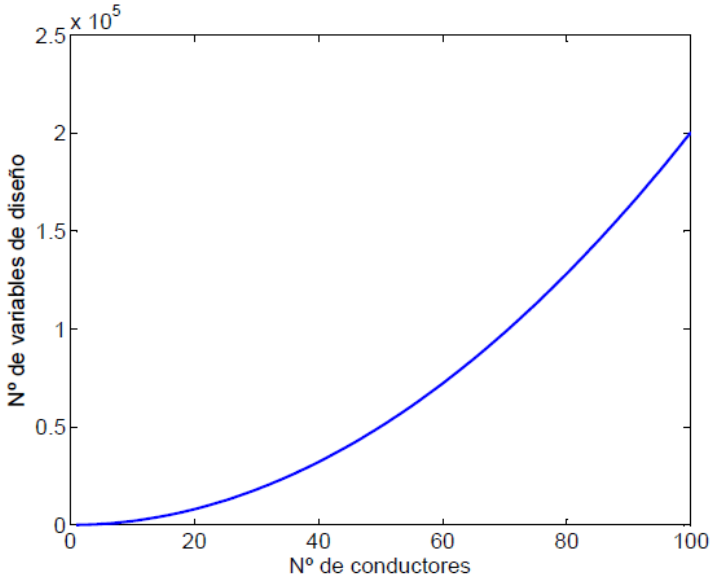


Figura 42.- Relación de variables de diseño y nº de conductores. Reducido

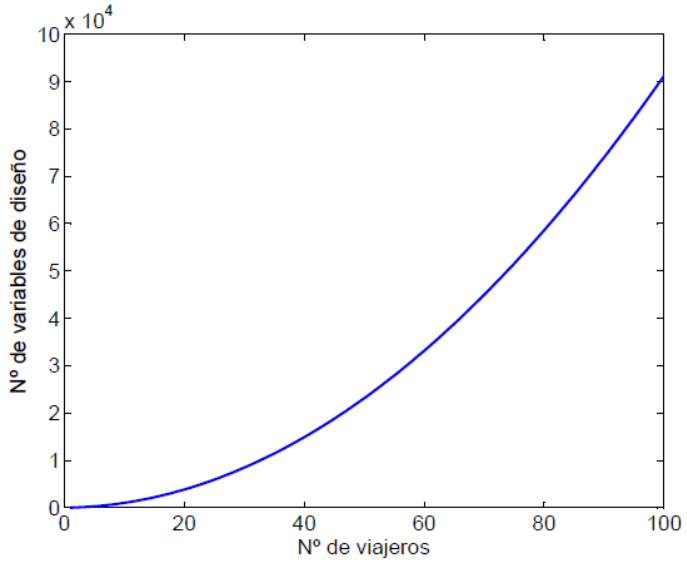


Figura 43.- Relación de variables de diseño y nº de viajeros. Reducido

La figura anterior muestra la función de número de elementos evaluada entre x = [0,100] e y = [0,100]. Se observa que la dependencia del número de elementos sigue siendo mayor con los conductores que con los viajeros, pero menor que el caso anterior. Para 100 conductores se generan del orden de $2 \cdot 10^6$ elementos. Esto implica una reducción de 5 veces menos elementos. Esta segunda opción es la que se implementará para reducir la limitación de memoria descrita anteriormente.

Capítulo 5

5. Simulaciones

En este capítulo se presentan los resultados de una batería de simulaciones de entornos. Se estudiarán una serie de parámetros para poder evaluar cual de los métodos tiene un mejor comportamiento, comparando sus resultados. También se realizará un estudio de sensibilidad de las diferentes variables, para ver su influencia sobre el resultado de las simulaciones.

Debido a que cada escenario es distinto, las variables de calibración del modelo y de los algoritmos serán distintas. Un estudio del Chicago Metropolitan Agency for Planning (CMAP) [11] [12] recoge la actividad de transporte en el noreste de Illinois en la que de 698 participantes, 469 optan por usar su propio vehículo mientras que el resto viaja en métodos alternativos de transporte. Se considerará esta proporción de conductores y viajeros representativa, por ello se va a realizar un estudio sobre una población de 30 vehículos y 15 viajeros manteniendo esta relación de conductores y viajeros (aproximadamente el 33% serán viajeros y el resto conductores).

5.1. Batería de pruebas

La batería de pruebas se divide en 3 partes. Un primer barrido de todos los parámetros con el método determinístico para analizar la relación entre los coeficientes de ponderación en escenarios similares. Estos escenarios constaran de un conjunto de 30 conductores y 15 viajeros participando en el *ridesharing*. Con la información de estas pruebas se podrán ajustar los coeficientes del modelo. Una segunda serie de pruebas resolviendo el mismo escenario con ambos métodos, demostrando la viabilidad de cada método en función del número de emparejamientos o la reducción de la distancia total recorrida. Por último, una tercera batería con el mejor método obtenido de las pruebas

anteriores, variando los posibles escenarios, para obtener un valor medio de distintas simulaciones.

El software desarrollado para las pruebas es una serie de bucles anidados que lanzan el algoritmo de optimización variando los parámetros del sistema, tal y como se explica a continuación:

```
for numero de viajeros entre M y N
   for numero de conductores entre X y Y
      for índice de ponderación entre I y J
          generación de entorno
          generación de usuarios
          generación de restricciones
          lanzamiento del algoritmo deterministico
          guardar resultados
      end for
   end for
end for
```

5.1.1. Primera batería de pruebas: Calibración del modelo

La primera batería de pruebas ejecuta el mismo escenario variando los distintos coeficientes del problema, el coeficiente de distancia recorrida (α) y el coeficiente de de emparejamientos (β). Para este escenario se emplean 30 conductores y 15 viajeros.

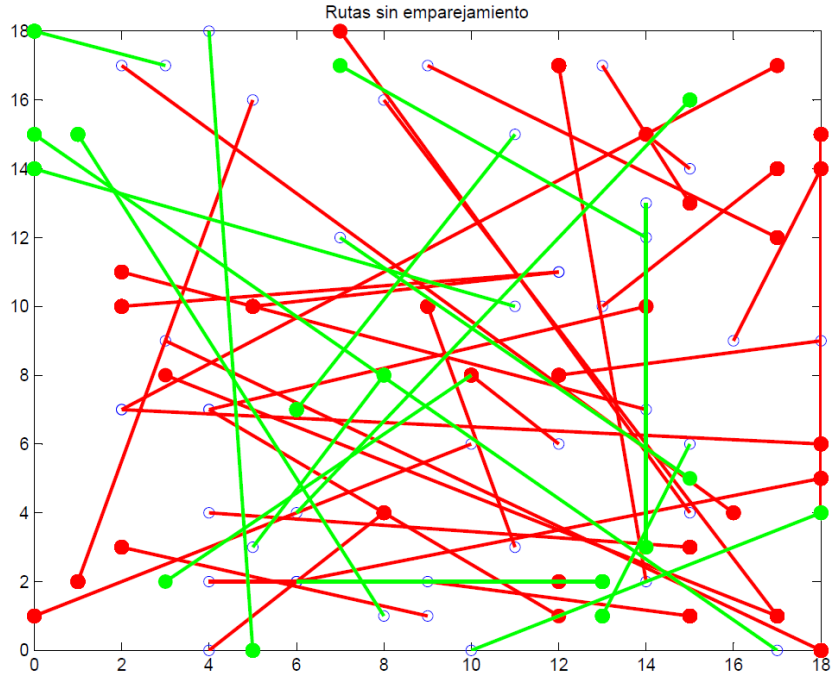


Figura 44.- Escenario de la primera prueba.

Estas pruebas servirán para configurar el modelo. Se ha realizado un análisis de sensibilidad de los dos coeficientes del sistema. Estos valores serán proporcionales a la distancia total del problema, dada la formulación de este. La distancia total de este primer problema es de 2381.318 unidades. Respecto al ajuste de los coeficientes, se pueden obtener los siguientes resultados:

- Manteniendo un valor fijo del coeficiente de emparejamientos (con un valor de 10) y variando el coeficiente α (de distancia) se alcana un valor constante de 3 emparejamientos para prácticamente todos los valores del coeficiente de distancia. Esto no implica que el número de emparejamientos sea el óptimo, ni que la distancia recorrida sea la menor. Establece una relación entre la distancia total y los emparejamientos.

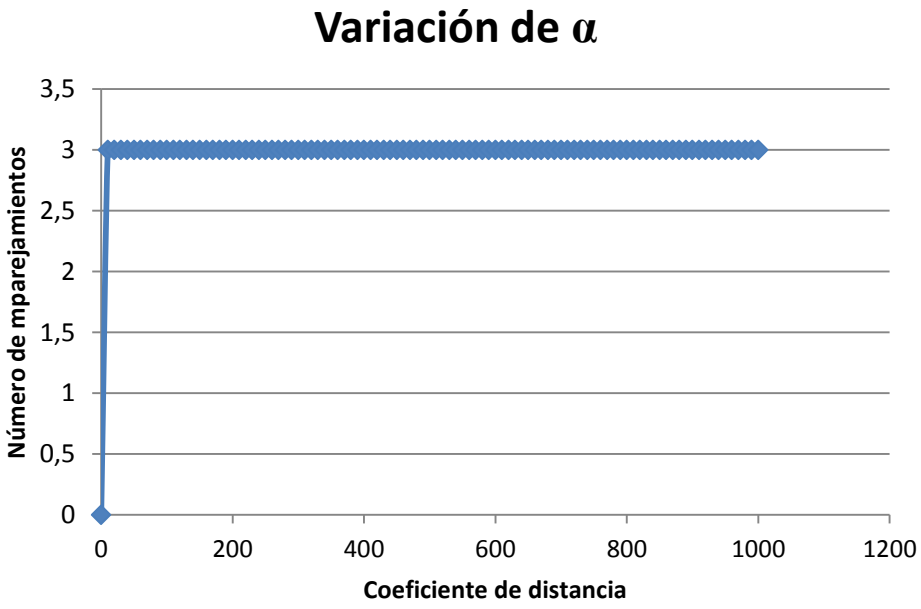


Figura 45.- Variación del coeficiente de distancia recorrida

- Manteniendo un valor fijo del coeficiente de distancia (con un valor de 10) y variando el coeficiente β se observa un incremento de los emparejamientos conforme aumenta este coeficiente, como se muestra en la Figura 46. Esto no implica que la distancia recorrida finalmente sea la óptima. El porcentaje de distancia reducida para los 15 emparejamientos es de un 9 % (2167.06 unidades recorridas finalmente respecto a las 2381.318 iniciales).

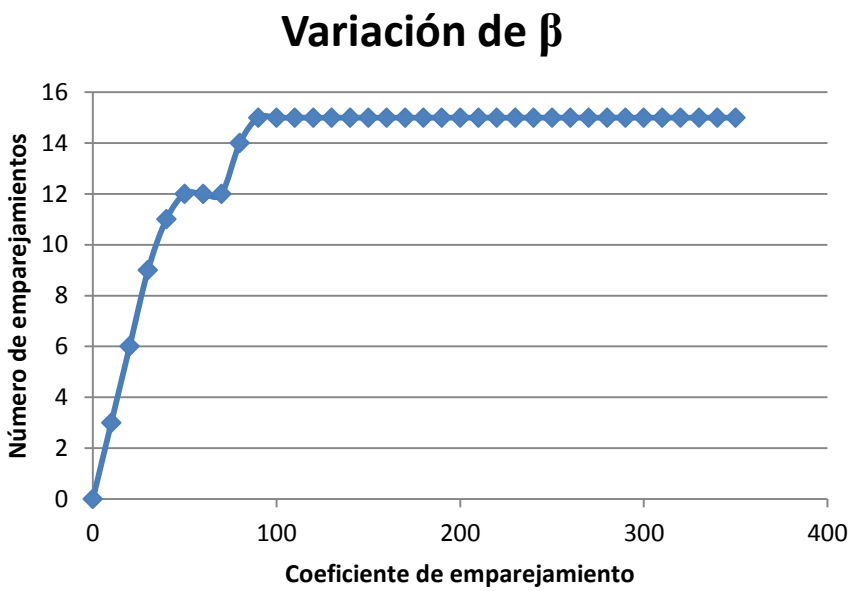


Figura 46.- Variación del coeficiente de emparejamientos

- Por último, se busca el valor óptimo del problema. Para este escenario se encuentra en el valor de $\alpha > 10$ y $50 \leq \beta \leq 70$. Para este intervalo se establecen 12 emparejamientos y la distancia recorrida se reduce en un 13%.

Para otro escenario distinto (Figura 47) el comportamiento del modelo es similar. La reducción de distancia es del 13.46%, valor similar al del caso anterior.

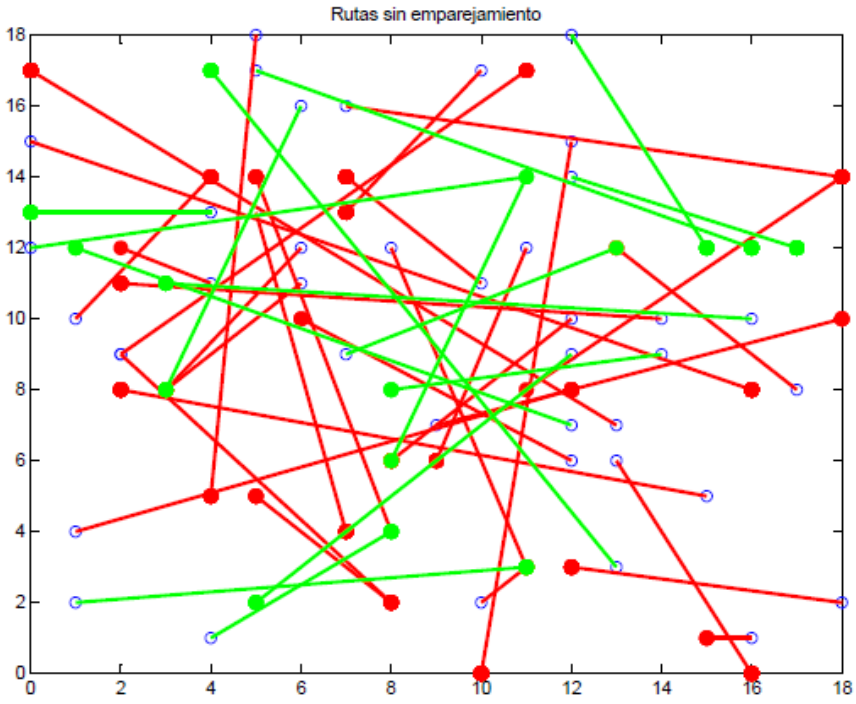


Figura 47.- Escenario alternativo para calibración

Para este escenario la distancia antes del *ridesharing* es de 1926.93. Como se muestra en la Figura 48, se produce un incremento que experimenta un pequeño

estancamiento en los 14 emparejamientos. Al igual que en el caso anterior, el máximo ahorro de distancia se produce en este intervalo. Este intervalo corresponde con valores de β entre 60 y 70.

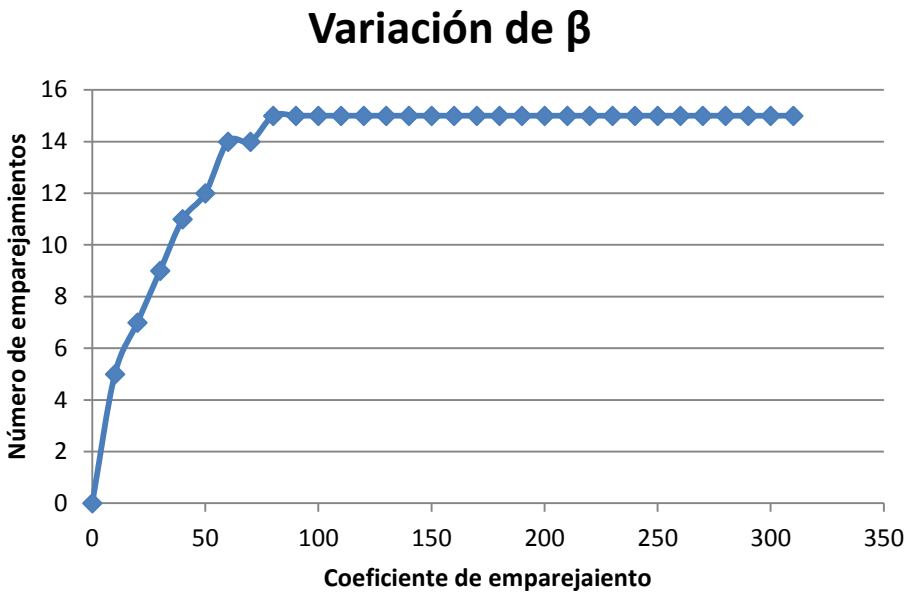


Figura 48.- Variación del coeficiente de emparejamientos. Escenario alternativo.

De las pruebas realizadas en este apartado, se puede afirmar que para esta proporción de participantes (30 conductores y 15 viajeros), un valor de β alrededor de 60 proporcionará un valor óptimo de reducción de distancia recorrida global. Para otras configuraciones, los coeficientes varían dependiendo de la distancia que recorran los participantes, lo que hace del ajuste otro problema de estudio. Para demostrar el efecto del *ridesharing* en la reducción de distancia recorrida, se realizará un estudio de distintos escenarios con el mismo número de participantes.

5.1.2. Segunda batería de pruebas: Selección del mejor método de optimización

Esta segunda prueba evaluará el funcionamiento de cada algoritmo. Dado que el *dynamic ridesharing* es un sistema que exige una respuesta temporal corta, un factor crítico será el tiempo de cómputo. Será por tanto interesante estudiar la evolución de este tiempo de cómputo para distinto número de usuario.

En la Figura 49 se recoge la relación de tiempos de procesamiento respecto a los conductores y a los viajeros definidos anteriormente para el algoritmo de optimización *bintprog*.

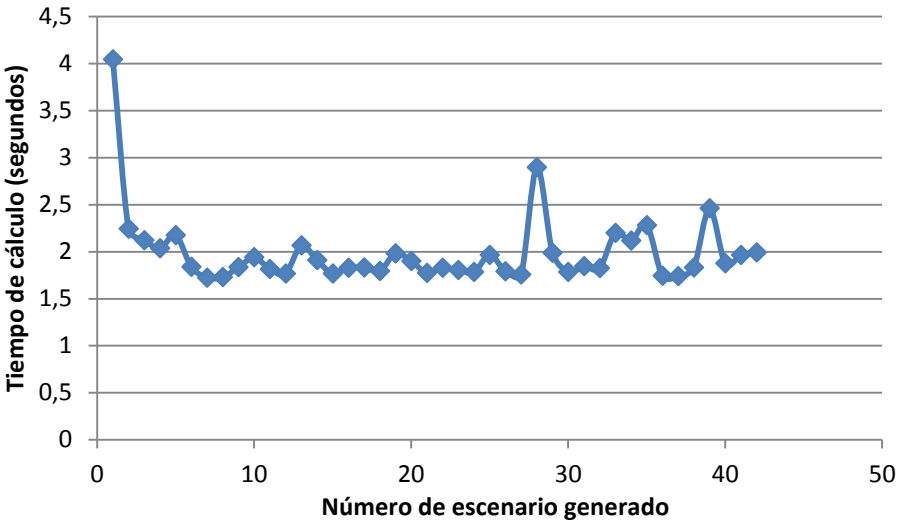


Figura 49.- Tiempo de cómputo para distintos escenarios con similar relación conductores/viajeros

Como se puede observar en la Figura 49 el algoritmo de *bintprog* proporciona unas respuestas temporales apropiadas para este tipo de servicio. Realiza los emparejamientos de manera correcta, proporcionando un ahorro de distancia final recorrida.

Por otro lado, el algoritmo genético responde de manera apropiada para escenarios con pocos participantes, pero su respuesta no es óptima. Se han probado distintas configuraciones del algoritmo para poblaciones mayores y menores. El resultado converge en menor tiempo para poblaciones menores, pero no converge siempre. Los resultados proporcionan distancias mayores que la inicial sin *ridesharing* debido a la convergencia prematura del algoritmo. Para poblaciones mayores los resultados mejoran, pero emplean demasiado tiempo (Figura 50 y Figura 51). Esto hace que se pierda la característica dinámica del *dynamic ridesharing.* La configuración de población, generaciones y demás factores es diferente para cada escenario, lo que hace que este método no sea el idóneo para este tipo de problemas.

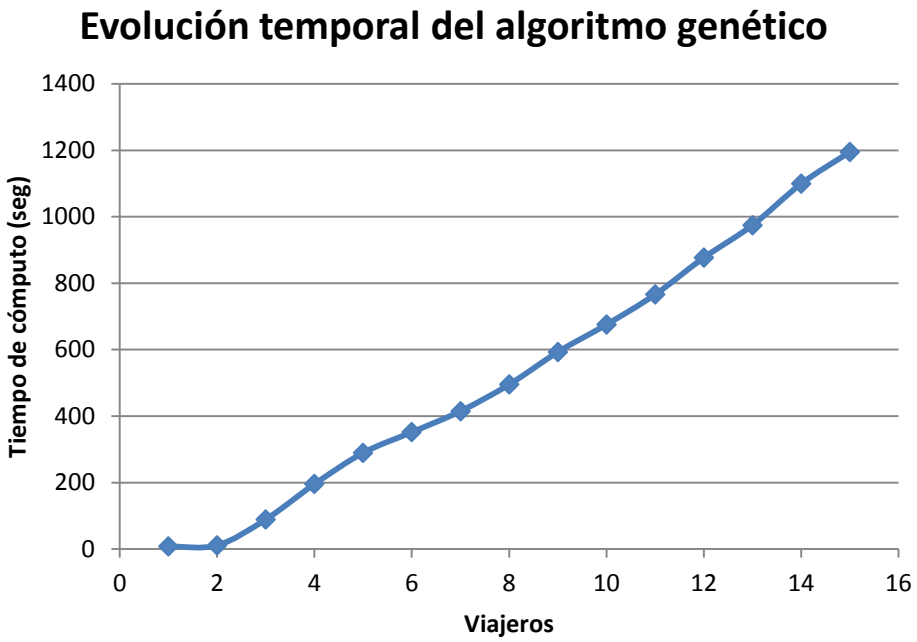


Figura 50.- Evolución temporal del algoritmo genético. Población de 15 conductores

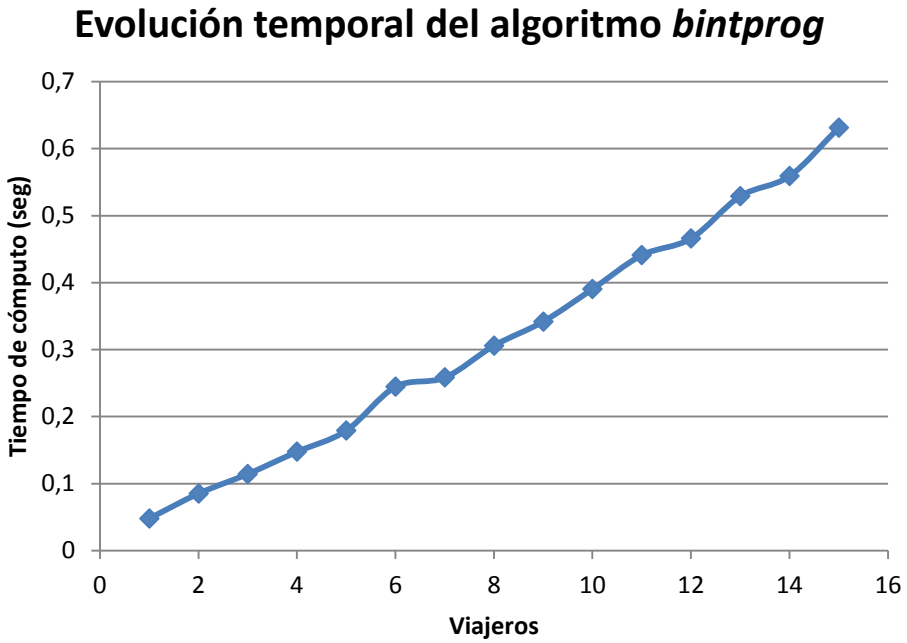


Figura 51.- Evolución temporal del algoritmo *bintprog*. Población de 15 conductores

Tras analizar los datos parece que el rendimiento del algoritmo *bintprog* es más adecuado que el algoritmo genético para este tipo de problemas.

5.1.3. Tercera batería de pruebas: Datos obtenidos

La tercera batería de pruebas resuelve una serie de escenarios aleatorios con el mejor método obtenido en la segunda batería de pruebas, el algoritmo *bintprog.* Los escenarios serán apropiados para la calibración obtenida en las primeras pruebas.

Tabla 7.- Simulación de escenarios para 30 conductores y 15 viajeros

Distancia inicial	Distancia optima	Tiempo de cálculo (seg)	Porcentaje de distancia reducido (%)
2144,02	1915,23	4,04	10,67
2418,74	2276,99	2,24	5,86
2355,59	2083,49	2,12	11,55
2270,17	2086,94	2,03	8,07
2025,39	1846,37	2,17	8,84
2136,65	2096,67	1,84	1,87
2231,80	1933,70	1,72	13,35
2270,52	2049,74	1,73	9,72
2188,57	2077,62	1,83	5,06
2375,53	2116,11	1,94	10,92
2353,09	2039,81	1,82	13,31
2176,35	1980,62	1,77	8,99
2076,10	1857,34	2,07	10,54
1920,43	1812,71	1,91	5,61
2466,94	2285,71	1,77	7,35
2345,94	2107,44	1,83	10,17
2431,93	2200,56	1,83	9,51
2264,81	2119,35	1,79	6,42
2167,11	1921,55	1,98	11,33
2097,29	1929,56	1,90	7,99
2141,70	1902,77	1,77	11,15
2241,04	2048,73	1,83	8,58
2256,39	2009,91	1,80	10,92
2254,12	2023,25	1,78	10,24
2364,92	2159,96	1,96	8,66
2034,64	1884,05	1,79	7,40
2208,07	2078,09	1,75	5,89
1872,91	1775,29	2,90	5,21
2111,85	1868,80	2,00	11,51
2083,76	1869,16	1,78	10,29
2174,00	2012,17	1,84	07,44
2238,39	1996,78	1,82	10,79
2215,17	1999,87	2,20	09,72
2350,66	2053,64	2,11	12,63

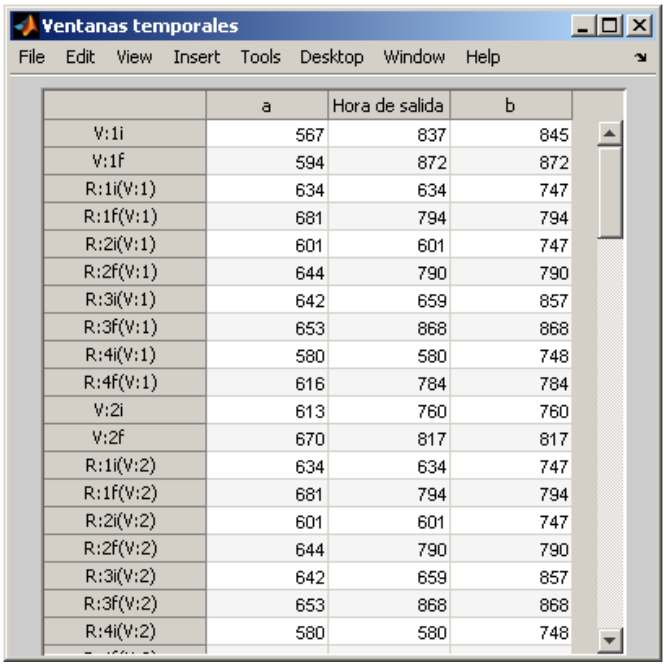
2321,90	2015,25	2,28	13,21
2169,54	2018,59	1,74	6,96
2351,65	2174,63	1,74	7,53
2454,25	2267,47	1,83	7,61
2364,81	1984,76	2,46	16,07
2105,70	1924,36	1,87	8,61
2101,39	1876,57	1,96	10,69
Valores medios			
2222,78	**2016,62**	**1,99**	**9,23**

De los valores de las simulaciones (Tabla 7) se observa que hay una reducción media de la distancia recorrida del 10 % aproximadamente, llegando en algunos casos al 16 %.

5.1.4. Prueba de ventanas temporales

Debido al bajo número de variables que acepta el software empleado, el método de las ventanas temporales se ve restringido a un bajo número de usuarios, por lo que la muestra sería muy reducida para poder apreciar los efectos del *ridesharing.*

Aún así, en la Figura 52 se observa la tabla resultado de la prueba, con tres columnas. La primera columna específica la hora mínima de salida/llegada de ambos (se ha convertido la hora y minutos únicamente a minutos para que el método pueda trabajar con esta variable como variable de tipo entero), conductor y viajero a cada nodo (i indica nodo de inicio y f nodo final). La segunda columna muestra la hora a la que debería llegar/salir del nodo. Se describe una hora para cada viajero viajando en cada vehículo, y cada conductor viajando de forma independiente. La tercera columna recoge la hora máxima de llegada/salida de cada nodo.



	a	Hora de salida	b
V:1i	567	837	845
V:1f	594	872	872
R:1i(V:1)	634	634	747
R:1f(V:1)	681	794	794
R:2i(V:1)	601	601	747
R:2f(V:1)	644	790	790
R:3i(V:1)	642	659	857
R:3f(V:1)	653	868	868
R:4i(V:1)	580	580	748
R:4f(V:1)	616	784	784
V:2i	613	760	760
V:2f	670	817	817
R:1i(V:2)	634	634	747
R:1f(V:2)	681	794	794
R:2i(V:2)	601	601	747
R:2f(V:2)	644	790	790
R:3i(V:2)	642	659	857
R:3f(V:2)	653	868	868
R:4i(V:2)	580	580	748

Figura 52.- Tabla de resultados temporales

El algoritmo realiza los emparejamientos adecuados siempre y cuando se realicen dentro de estas ventanas temporales. El método no acepta más de 5 conductores y 4 pasajeros, lo que hace que las ventanas temporales no sean abordables con las herramientas actuales.

6. Resultados

Se ha llegado a la conclusión de que el algoritmo genético funciona de manera apropiada para pocos participantes, pero la respuesta temporal de éste empeora conforme aumenta el número de participantes. El algoritmo genético es de gran utilidad cuando se requiere una solución BUENA, pero no cumple con los criterios de resultado óptimo. Este tipo de algoritmos puede tardar bastante, en circunstancias de gran número de participantes, en converger lo que hace que este método quede relegado a un segundo puesto. Pueden converger prematuramente debido a una serie de problemas. Si un individuo que es más apto que la mayoría de sus competidores emerge muy pronto en el curso de la ejecución, se puede reproducir tan abundantemente que merme la diversidad de la población demasiado pronto, provocando que el algoritmo converja hacia el óptimo local que representa ese individuo, en lugar de rastrear el paisaje adaptativo lo bastante a fondo para encontrar el óptimo global. Esto es un problema especialmente común en las poblaciones pequeñas, donde incluso una variación aleatoria en el ritmo de reproducción puede provocar que un genotipo se haga dominante sobre los otros [13]. Al aumentar la población, aumenta el tiempo de cómputo por lo que el modelo de transporte se ve comprometido en su característica dinámica.

En cambio, el algoritmo de *bintprog* proporciona un resultado temporal adecuado y los resultados para un escenario basado en proporciones reales representan una reducción media del 10% de la distancia recorrida en los escenarios. Esta cifra puede no parecer excesivamente elevada, pero aplicado a las cifras de transporte, los efectos positivos, tal y como se muestra a continuación, se multiplican.

Tabla 8.- Factores de emisión de distintos tipos de vehículo [g/kg de combustible][14]

Categoría	CO	NO_x	NMVOC	CH_4	PM	CO_2
Gasolina	75,99	10,89	13,44	1,19	0,03	3,18
Diesel	3,77	11,12	0,61	0,07	0,8	3,14
Autobuses	9,82	34,84	3,06	0,38	1,34	3,14
Motocicletas	577,51	3,63	165,71	5,12	3,68	3,18

Tabla 9.- Kilómetros recorridos anualmente en España por un vehículo particular [15]

	Vehículo medio
TOTAL (km)	12.562,9

Se evaluarán a continuación los beneficios medioambientales y económicos del *ridesharing* suponiendo un consumo medio de 4 litros a los 100 km para un vehículo diesel y un consumo medio de 7,5 litros a los 100 km para un vehículo gasolina [16], y un parque automovilístico [17] de una ciudad (los datos corresponden con los de la ciudad de Madrid) en el que el 44% son vehículos gasolina (1.433.210 vehículos) y el 56% restante son diesel (1.856.978 vehículos). Para el cálculo del ahorro económico se ha considerado un precio de diesel de 1,335 euros y de gasolina de 1,424 euros. Los resultados se recogen en la Tabla 10.

Tabla 10.- Comparativa entre el método convencional de transporte y la alternativa propuesta. [Toneladas emitidas]

Modo de transporte	CO	NO_x	NMVOC	CH_4	PM	CO_2	Precio (M €)
Sin ridesharing	56565,99	15609,55	9964,29	893,6	588,64	4482,16	3168,73
Con ridesharing	50909,40	14048,6	8967,87	804,26	529,77	4033,94	2851,86
Comparativa	5656.59	1560,95	996,42	89,33	58,87	448,22	316,87

La reducción de contaminantes queda patente, y aunque el método no elimina totalmente las emisiones, cualquier reducción de contaminantes es un buen resultado en cifras "medioambientales". Este método proporciona un ahorro anual de 316,87 millones de euros en concepto de ahorro de combustible.

Capítulo 6

7. Conclusiones

Como se ha visto en los dos primeros capítulos, se ha definido el modo de transporte denominado ridesharing y se han descrito todas sus variantes. Dada la complejidad que deriva de las distintas modalidades, se ha optado por la resolución de la variante en la que un pasajero puede ser emparejado por un único conductor y este a su vez solo puede recoger a un viajero. El proyecto se ha centrado en la recreación del modelo de *ridesharing.*

Se ha formulado el un modelo matemático que cumple con el funcionamiento descrito del *ridesharing,* comprobando su correcto funcionamiento. Para ello se realizan una serie de pruebas que prueban el modelo en escenarios artificiales.

Se han comparado distintos métodos de optimización, optando finalmente por el que menor tiempo de cómputo ha demostrado, ya que el tiempo era un factor importante del problema que lo dota de la característica dinámica. Este método garantiza que, para este problema, el resultado obtenido es óptimo.

De las simulaciones realizadas se han obtenido una serie de datos favorables que demuestran la viabilidad de este modelo de transporte. Se ha realizado un análisis de sensibilidad para calibrar el modelo. Los distintos coeficientes necesitan estar relacionados el uno con el otro y deben ser directamente proporcionales a la distancia total del problema. Se han representado los resultados en distintas unidades, como puede ser el porcentaje de distancia que se ahorra o el número de emparejamientos lo que permite compararlo con otros modos de transporte.

Con todo lo expuesto, se puede considerar que el *dynamic ridesharing* es válido como modo alternativa al modelo tradicional de transporte privado. Sin embargo, su viabilidad tecnológica, y en consecuencia su rentabilidad, depende de que varios aspectos que se mencionan a continuación encuentren solución.

8. Trabajos futuros

El problema se ha acotado al modelo estudiado, pero el estudio de las otras variantes del modelo podría mejorar el alcance de los resultados obtenidos en este trabajo. En efecto, es de esperar que las variantes más complejas del problema aporten mejoras significativas a los resultados expuestos en este trabajo, debido principalmente a que al aumentar la flexibilidad del sistema aumentarán las opciones de emparejamiento. En el trabajo no se han considerado condiciones temporales, como intervalos de recogida y de entrega, que aumentarían también la flexibilidad, y en consecuencia la aceptación del *ridesharing*. Esto añadiría otro grado de dificultad al problema, pero podría ser más atractivo para los usuarios.

Otra línea de trabajo podría centrarse en el empleo de algoritmos específicos para la solución de este, o el empleo de algoritmos basados en fenómenos biológicos. Se ha probado el algoritmo genético, viendo que el tiempo de cómputo no se ajustaba a lo deseado para este tipo de servicios. Se podrían considerar alternativas como algoritmos de inteligencia de enjambres, basados en agentes que se comportan como ciertos organismos, como por ejemplo las hormigas o las abejas [18]. Otra alternativa sería trabajar en la solución del modelo con ventanas temporales mediante otro tipo de algoritmos MINLP.

Por último, se podría considerar el desarrollo de la red de *ridesharing* creando una aplicación móvil de localización GPS con conexión a un servidor en el que se ejecute el algoritmo de optimización y realizar una prueba real. Esto mostraría las posibles barreras que tendría que afrontar una empresa que ofreciera este servicio, como podría ser la obtención de masa crítica, o los costes derivados del desarrollo y mantenimiento del sistema.

Referencias

[1]. Asignatura del Máster en Ingeniería Industrial de la UC3M: Fuentes de energía. "Capítulo 4: Petróleo. Combustibles líquidos y gaseosos. Usos y precios. Conversión." Mathieu Legrand, 2013.

[2]. "Informe de emisiones de Gases de Efecto invernadero en España 1990 - 2012". Informe 2013. WWF. Consultado en 2014 : http://awsassets.wwf.es/downloads/informe_de_emisiones_de_gei_en_espana_1990_2012.pdf

[3]. "Los problemas del coche" Ecologistas en acción. Consultado en 2014: http://www.ecologistasenaccion.es/article9846.html

[4]. "Informe SMART 2020." The Climate Group. Consultado en 2014: http://www.smart2020.org/_assets/files/02_Smart2020Report.pdf

[5]. "Los beneficios de compartir coche" Amovens. Consultado en 2014 : https://www.amovens.com/es/beneficios-compartir-coche

[6]. "Solving the Ridematching Problem in Dynamic Ridesharing". Tesis doctoral. Wesam M. A. Herbawi. 2012.

[7]. "Deterministic Methods for Mixed Integer Nonlinear Programming". Sven Leyffer. University of Dundee. Tesis Doctoral.Consultado en 2014: http://www.mcs.anl.gov/~leyffer/papers/thesis.pdf

[8]. "Undercover - A primal MINLP heuristic exploring a largest sub-MIP". Timo Berthold, Ambros M. Gleixner.g

[9]. "A vehicle routing model with Split delivery and stop nodes". Lonaro Berbotto, Sergio García y Francisco J. Nogales. Departamento de Estadística de la UC3M. Abril de 2011

[10]. OPTI Toolbox – NOMAD. Consultado en 2014: http://www.i2c2.aut.ac.nz/Wiki/OPTI/index.php/Solvers/NOMAD

[11]. "The ridematching problem with time windows in dynamic ridesharing: A model and a genetic algorithm". Wesan Herbawi y Michael Weber. Institute of Media Informatics. University of Ulm, Germany.

[12]. "Travel Tracker Survey" Chicago Metropolitan Agency for Planning (CMAP). Consultado en 2014: http://www.cmap.illinois.gov/travel-tracker-survey

[13]. "Algoritmos genéticos" Jorge Arranz de la Peña, Antonio Parra Truyol. Universidad Carlos III de Madrid.

[14]. "Group 7: Road Transport" European Enviromental Agency. Consultado en 2014: http://www.eea.europa.eu/publications/EMEPCORINAIR5/page016.html

[15]. "Encuesta de hogares y medioambiente 2008: Km recorridos al año por vehículos para uso personal". Instituto Nacional de Estadística (INE). http://www.ine.es/jaxi/tabla.do?path=/t25/p500/2008/p10/l0/&file=10020.px&type=pcaxis&L=0

[16]. "Consumo de combustibles y emisiones de CO_2 " Insitituto de Diversificación y Ahorro de la Energíe (IDAE) http://coches.idae.es/portal/BaseDatos/MarcaModelo.aspx

[17]. "Parque de vehículos por tipos, ccaa, provincias y carburantes" Dirección general de tráfico. Consultado en 2014: http://www.dgt.es/es/seguridad-vial/estadisticas-e-indicadores/parque-vehiculos/prov-y-tipo-carburante/

[18]. "Swarm intelligence systems for transportation engineering: Principles and applications" D. Teodorovic. Transportation Research Part C.

9. Anexo

9.1. Características del equipo LG E500-J.AP51B

Figura 53.- Característica del equipo empleado